Processing of Mars Exploration Rover imagery for science and operations planning

Douglass A. Alexander, Robert G. Deen, Paul M. Andres, Payam Zamani, Helen B. Mortensen, Amy C. Chen, Michael K. Cayanan, Jeffrey R. Hall, Vadim S. Klochko, Oleg Pariser, Carol L. Stanley, Charles K. Thompson, and Gary M. Yagi

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, USA

Received 15 April 2005; revised 17 October 2005; accepted 26 October 2005; published 4 February 2006.

[1] The twin Mars Exploration Rovers (MER) delivered an unprecedented array of image sensors to the Mars surface. These cameras were essential for operations, science, and public engagement. The Multimission Image Processing Laboratory (MIPL) at the Jet Propulsion Laboratory was responsible for the first-order processing of all of the images returned by these cameras. This processing included reconstruction of the original images, systematic and ad hoc generation of a wide variety of products derived from those images, and delivery of the data to a variety of customers, within tight time constraints. A combination of automated and manual processes was developed to meet these requirements, with significant inheritance from prior missions. This paper describes the image products generated by MIPL for MER and the processes used to produce and deliver them.

Citation: Alexander, D. A., et al. (2006), Processing of Mars Exploration Rover imagery for science and operations planning, *J. Geophys. Res.*, *111*, E02S02, doi:10.1029/2005JE002462.

1. Introduction

[2] The successful landing of NASA's Mars Exploration Rover (MER) rovers "Spirit" and "Opportunity" on the surface of Mars in January of 2004 marked the beginning of a mission that was highly dependent on the timely and efficient ground processing of engineering and science camera images. Rover instrument data telemetered daily to the MER Ground Data System (GDS) at the Jet Propulsion Laboratory (JPL) had to be processed with sufficient turnaround for use by operations and science personnel who convened in tactical planning sessions to decide the following day's rover activities.

[3] The Multimission Image Processing Laboratory (MIPL) at JPL was tasked with processing the MER instrument telemetry data on the GDS as part of the Operations Product Generation Subsystem (OPGS). The scope of MIPL's role was comparable to those served in 1997 and 2001 in support of operations for the Mars Pathfinder (MPF) [*LaVoie et al.*, 1999] and Mars Polar Lander (MPL) missions, respectively. Similarities in mission objectives with MPF and MPL enabled much reuse of code and algorithms in the development of the MER product-generating application software.

[4] For MER, MIPL's capabilities were integrated into the critical path of tactical operations more so than with the previous Mars lander missions. As such, MIPL designed and implemented an automated end-to-end data product generation system that relied on the ability to transport in sequence, or "pipeline", the rover instrument data between disparate application processes running concurrently on GDS computing resources.

[5] A high level overview of the MIPL product generation system is diagramed in the context of the MER GDS in Figure 1. It illustrates the ingestion of rover instrument telemetry data into the MIPL system to produce labeled first order products called Experiment Data Records (EDRs) for use by science teams. In the case of camera instrument data, the data was processed further into a plethora of derived data products called Reduced Data Records (RDRs) for the extraction of terrain information valuable for planning rover navigation. The derived products included (1) single image types such as geometrically corrected images, (2) stereo-image types characterizing the scene terrain, and (3) multiple-image types including unified 3-dimensional terrain models and multi-image mosaics. Each resultant product was placed onto the mission's Operations Storage Server (OSS), a secured file server, for retrieval by a variety of customers that primarily included teams of rover navigation planners and science activity planners. Finally, products were delivered outside the MER GDS environment to the home institutions of science teams and other remote sites using the File Exchange Interface (FEI), a file delivery application discussed in more detail in section 7.3. Products were also accessible through interface with Planetary Data System (PDS) Web-based resources such as the Image Atlas managed by the Image Node at JPL and the Analyst's Notebook managed by the Geosciences Node at Washington University in St. Louis.

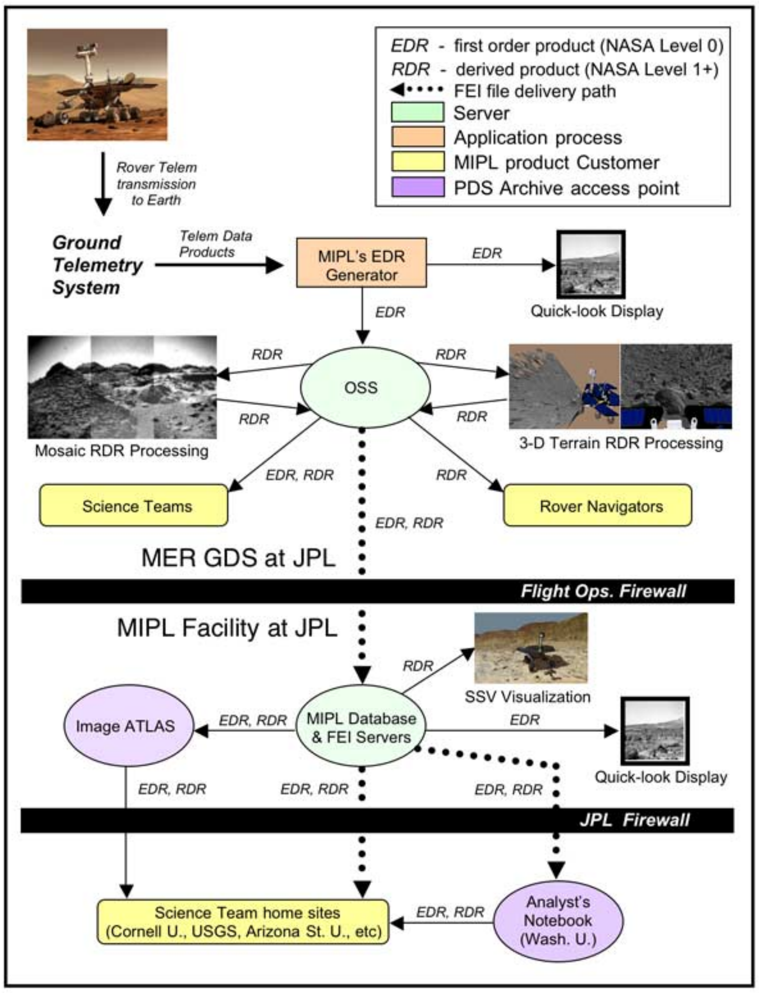


Figure 1. MIPL system overview in MER GDS context.

[6] This paper will elaborate on the processes, capabilities, products and customers shown in Figure 1, and will include discussion of the MIPL pipeline's underlying system architecture, design and operation. Though the focus will be on the OPGS products generated by MIPL, it should be noted that the Panoramic Camera (pancam) and Microscopic Imager (MI) teams also generated camera image products onto the OSS as science products separate from the OPGS products.

2. System Architecture

[7] The MIPL operations system was composed of a suite of individual application programs that performed distinct operations on data products. The application processes were managed by higher level "glue" software that determined the identity of process inputs and the order of program execution. The integration of the application programs with the "glueware" established the complete pipeline system.

2.1. Operational Environment

[8] The preferred systems for MER operations consisted of a mix of Sun-Solaris and Intel-Linux machines using a Sun Network File System (NFS) system. MIPL opted to go with four Linux machines per rover mission, each with dual processors running at 2GHz with 1GB of RAM and connected via a high speed optical link to the OSS shared file system. The OSS was an elaborate directory structure that generally organized data file storage by the categories of mission phase, Mars solar day (Sol) and instrument type. Symbolic links defining file pointers were used to expedite data processing on the OSS. The OSS physically resided on an NFS server, providing access to all workstations networked within the MER flight Local Area Network (LAN). The OSS was then replicated for each rover, with a full redundant backup, for a total of four distinct systems.

2.2. Customers

[9] The many products generated by MIPL were needed by several teams for rover operations planning. Among them were the science teams who used the Science Activity Planner (SAP) tool [*Norris et al.*, 2005] for targeting surface features of interest as part of short-term planning, rover tactical operations planners who used the Rover Science Visualization Planner (RSVP) tool [*Maxwell et al.*, 2005] in planning the rover maneuver command sequences for the next Sol's activities, the rover mobility team that was responsible for reviewing image data to determine where the rover actually had moved in comparison to the nominal traverse plan for the previous day, and the rover long-term

planners who analyzed multi-image mosaics to plot the course of rover movement several days in advance.

[10] Three additional customers who used the MIPL products were remotely located science teams that evaluated the data more extensively outside the tactical timeline, the Educational Public Outreach office who posted data to the Mars Rovers public Web site and distributed it to hundreds of museums across the United States, and the Public Information Office (PIO) which disseminated information during press conferences.

2.3. Performance Requirements

[11] MIPL's data processing requirements were directly affected by the large quantity of instrument data projected to accumulate on the GDS each day as part of the tactical rover operations. For contrast, the MPF data volume was populated mostly by 256×256 pixel sized images acquired by the lander's stereo cameras, while each MER rover had nine cameras that acquired images that typically were 1024×1024 pixels in size. The volume of data returned by a single rover from one Site was often more than MPF returned for its entire mission. All of this amounted to an enormous planned data volume that had to be managed efficiently by MIPL's system.

[12] The most challenging requirements were related to the time constraints by which data products were needed for use in the rover tactical planning meetings. Three important timing requirements follow:

[13] 1. EDRs, or the NASA Level-0 data products, had to be generated in near real-time. In other words, nearly as fast as the rate at which the rover was capable of transmitting the data to Earth either directly or via orbiter relay. Given that telemetry volume was planned for about 200 EDRs per downlink, this requirement would keep processing at a pace compliant with the timeliness requirements set for the 3-D terrain end products, thereby allowing sufficient time for the daily rover navigation planning session.

[14] 2. The 3-D terrain meshes needed for rover and robotic arm planners had to be generated within 1 hour from the end of the applicable downlink session, a turnaround necessary to allocate sufficient time for the subsequent daily rover planning session.

[15] 3. All products generated on the OSS had to be available to flight LAN users upon creation and distributed to remote sites external to the flight LAN within seconds.

2.4. Other Requirements

[16] There were other indirect requirements that shaped the design and operation of MIPL's system. For example:

[17] 1. The target hardware platforms and their configurations were unknown when the design of MIPL's software capabilities began. At the time, the project considered several options. As a result, MIPL designed and implemented its system in such a way that it could run in either Sun-Solaris or Intel-Linux environments. This allowed other UNIX-like systems or environments, including a variety of Linux systems, to be accommodated with relative ease.

[18] 2. An early decision by the mission restricted the use of any Data Base Management System (DBMS), such as SYBASE, MySQL, etc., from being in the critical path of operations. The decision was based on concerns that failure of a DBMS might jeopardize mission planning for one or more Sols, and that DBMS failover was too costly.

[19] 3. Use of a Web server was also restricted inside the flight LAN.

[20] 4. The use of Planetary Data System (PDS) image and metadata formats, common standards that are regularly applied in the use of planetary data, were mandated by the project for all operational and archived image data.

2.5. Heritage From Prior Missions

[21] The reuse of software available from previous missions similar to MER, such as MPF and MPL, provided MIPL with a genuine opportunity for cost savings. This came principally in two forms: (1) software that generated the RDR, or NASA Level 1+ derived product, and (2) the Video Information Communication And Retrieval (VICAR) file format.

2.5.1. RDR-Generation Software

[22] The software used for RDR generation was originally developed for MPF, starting in 1994. The software worked, but it was specific to MPF, with hardcoded constants, inflexible algorithms, and much duplication of code.

[23] During the development phase for MPL, it was realized that the MPF code could be reused with some modifications. Furthermore, future missions would have similar requirements. MIPL developers therefore embarked on an effort to reengineer the software to make it reusable in future missions [*Deen*, 2003a].

[24] The result was a set of application programs without mission-specific references. All mission-specific code is encapsulated into a library known as Planetary Image Geometry (PIG). This C++ library uses object-oriented abstractions to hide the mission-specific code behind a common interface. The common interface provides facilities to manipulate things like camera models, camera pointing, surfaces, coordinate systems, image files and metadata. For example, a generic pointing model can repoint cameras on the basis of a specified set of pointing parameters. The logic in MER-specific subclasses apply MER kinematics to point the pancam, Navigation Camera (navcam) or MI on the basis of MER parameters, and return a pointed camera model.

[25] The PIG library currently supports six "missions": MPF, MPL, Mars '01 testbed, a JPL testbed rover called Field Integrated Design & Operations (FIDO), MER, and a "generic" mission. Adaptation times for each mission have ranged from a few days in the Mars '01 testbed case to a few months in the MER and FIDO cases. MER required both adaptation and the development of many new capabilities to be added to the library. Examples are the ability to support multiple rover locations (Sites), as well as support for fisheye-lens camera models used by the Hazard Avoidance Cameras (hazcams) as described by *Gennery* [2002]. These new capabilities added for MER will be available to future missions, as well as for reprocessing data from old missions if desired.

[26] This legacy code reuse has worked well in this area. The mission-independent application code is about 2.5 times larger than the PIG library. Each mission's code represents less than 5% of the entire code base, sometimes much less [*Deen*, 2003a], which translates into significant cost savings for MER and future missions.

2.5.2. VICAR and PDS File Formats

[27] The MER project required that all science instrument products, including image products, be generated using the PDS file format. However, the legacy software described in the previous section was written to use the VICAR file format, which was developed at MIPL and had been used for decades by MIPL software (see http://www-mipl.jpl.nasa.gov/vicar). To preserve the considerable software heritage, a "dual-labeled" file format was designed so that both a PDS and a VICAR label were attached to the images. The VICAR I/O system was modified to recognize and skip over the PDS label on read, while the application programs would write pure VICAR format with a separate "transcoder" program to add an equivalent PDS label afterward. The PDS format already supported a mechanism to skip over the embedded VICAR label. So, the dual-label design accommodated MIPL software written to use only the VICAR label as well as software developed outside of MIPL written to use only the PDS format. This had the additional advantage of making available the large base of legacy image processing applications written for VICAR.

[28] As a note, the only significant part of the VICAR label that was not duplicated in PDS (and vice versa) was the history label. On occasion, access to this label added to the analysis of the image data, since it listed the parameter values used by programs when the data was processed.

3. Pipeline Design and Implementation

[29] The need for camera images to be processed and delivered in a compressed time frame induced MIPL to develop a product generation and delivery pipeline that was robust and almost completely autonomous in its operation. This section presents a brief overview of the pipeline's design and identifies the interconnecting process streams; see *Alexander et al.* [2005] for more details.

[30] The pipeline's glueware was developed as a script roughly 10,000 lines of code in length using the Unix Bourne shell [*Kochan and Wood*, 1998]. Development was driven by a few factors: (1) scripts are generally more flexible and easier to maintain than low level C++ code, (2) the MER GDS provided a Unix-based environment that supported the Sun-Solaris and Linux Red Hat operating systems, making the glueware script readily deployable on multiple systems, (3) establishing Unix as the product generation system baseline, combined with familiarity of Unix amongst operations team members, made the design amenable to future integration of ancillary tools developed as scripts during mission operations, and (4) heritage brought from MPL, where the product generation system was developed under the Bourne-again shell (Bash), provided precedents in a few areas of the pipeline design (such as driving processing events based on file detection).

[31] At the outset, the design was affected by project policy reflected in two requirements mentioned in section 2.4: (1) the restricted use of a database in the critical path of mission operations, and (2) the restricted use of a Web server inside the flight LAN. As a result of the first restriction, the pipeline was devoid of a database. The second restriction limited the distribution of the system's data product tracking results.

[32] Without the capability of database stored procedures that trigger and terminate process I/O automatically, the design's fundamental strategy became one of driving each processing event on the basis of detecting files on the OSS file system. The event-driven processing involved the continuous search for data products on the OSS that matched specific criteria. The returned status of each attempted search drove the execution of subsequent events that were scheduled as segments within a sequential "stream" of processes. The pipeline performance was a function of two design factors: (1) the system's ability to spawn process streams in parallel automatically and (2) the ability of operators to distribute the streams manually across available machine resources for load balancing.

[33] Briefly, each process stream consisted of an endless loop that continuously searched temporary directories on the OSS that served as "queues" for processing of data products. To speed up file handling, files were symbolically linked, or "soft linked", so that in each case only the address of the file's pointer was changed instead of copying the entire file from one queue to another. This avoided the problem of accessing partially written files from other processes. There were also other temporary directories for contingency purposes, including the backup of each input file's link and storage of links for files that failed application processing.

[34] The pipeline managed five independent processing streams that were executed in parallel to one another: (1) the Application Stream, which invoked the application program's command line for each input file, (2) the PDS Labeling Stream, which converted the VICAR-formatted output to a dual-labeled PDS-compliant file using a Java "transcoder" program, (3) a Product Delivery Stream for transport of the final PDS-labeled products to the nominal customer directories on the OSS, (4) a second Product Delivery Stream for transport of the final PDS-labeled products to remote sites external to the flight LAN via FEI, and (5) the Image Display Stream, which called the Java EDR Display Interface (JEDI) for image display of each EDR file onto a user-specified monitor.

[35] Importantly, the pipeline was flexible on two fronts: developmentally and operationally. Regarding the former, the design was pliable enough for the quick addition of new capabilities ad hoc during MER operations to accommodate growth in mission requirements. As an example, the evolution of the new Slope RDR product type (see section 4.2.2), from design to product generation, took just over three days, with the majority of time committed to modifying application software. Integration of the capability into a test pipeline and subsequent systematic testing was performed in just over one day. Final activation was completed during the daily 20-minute shutdown/restart of the operational pipeline that was practiced to manage temporary file growth on the OSS.

[36] Operationally, the flexibility was evidenced by a special mode of pipeline invocation wherein the user controlled the data product flow into private directories without touching the OSS. This was useful in generating

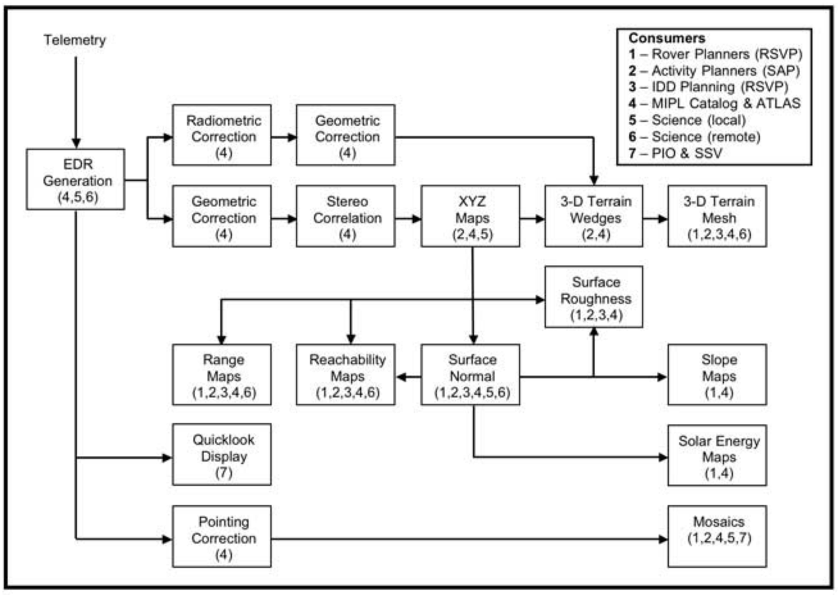


Figure 2. MIPL product generation pipeline data flow.

nonstandard data products for special purposes at the request of a customer, and for testing new processing methods without impacting MER operations.

4. Pipeline Data Products

[37] The myriad products generated by the MIPL automated pipeline are shown in their sequence of processing in Figure 2. They are described in this section at a fairly high level, but with enough detail to understand what they are. For full details on these products, including file formats, metadata contents, and algorithm references, see *Alexander et al.* [2003].

4.1. First Order Products (EDRs)

[38] Each MER rover payload instrument acquired unique data that were saved onboard as separate products. Upon transmission to Earth, the products were each split into parts and packaged inside telemetry packets. Each packet was identified according to the type of data it carried, plus additional ancillary information required for data product reconstruction on the ground.

[39] The ground data subsystem that was upstream of MIPL in the GDS configuration reconstructed the telemetry packets into data products, with each made up of two components: (1) a binary file containing the instrument data and (2) a meta-file containing ancillary information that characterized the instrument data. Additionally, the upstream subsystem was capable of handling some bit level errors that were due to transmission noise. When such corrections were not possible, or when data were missing, the meta-file was tagged with information flagging the condition. In most cases, corrupt or partial data products were retransmitted and when new versions were available, they were used to overwrite the older, less complete versions on the GDS.

[40] MIPL digested the telemetry data product to create the first order, or “raw”, EDR product that corresponds to Level 0 in the NASA data processing hierarchy. Each MER EDR contained the instrument data reformatted into a usable product, plus a complete label that was fully compliant with PDS rules and guidelines, making the EDR archive-ready. See Figures 3a and 3b for examples of camera instrument EDRs. For all MER camera instruments, which included navcam, pancam, MI and two hazcams, the resultant EDR files contained an attached label. A discussion of the nonimage instrument data processing is beyond the scope of this paper.

4.2. Derived Products (RDRs)

[41] Once the image EDRs were generated, a wide variety of products (mostly images) were automatically and systematically generated from them. These RDRs can be broken down into three broad classes: those derived from a single image, those derived from a stereo image pair, and those making use of multiple images. These products are described below. The 3-letter identifiers specified in parentheses after each type appeared in the filenames to distinguish these products uniquely, and are provided for reference (for filename conventions and table of identifiers, see *Alexander et al.* [2003, section 4.4]).

[42] The algorithms used for radiometric correction, linearization and XYZ derivation were substantially similar to those used for MPF [*LaVoie et al.*, 1999], with incremental improvements (such as the addition of the CAHVORE camera model). Disparity maps and terrain meshes were also created for MPF, but the algorithms used for MER were significantly different.

4.2.1. Single-Image Products

[43] These are all derived from a single EDR.

[44] Inverse LUT (ILF, INN, ISF): The cameras produced 12-bit images. A common compression method, especially for pancam and MI, was to use an onboard look-up table (LUT) to convert this to 8-bit data for transmission. An inverse LUT process restored the original 12-bit data format, with some loss of intensity resolution. If the EDR

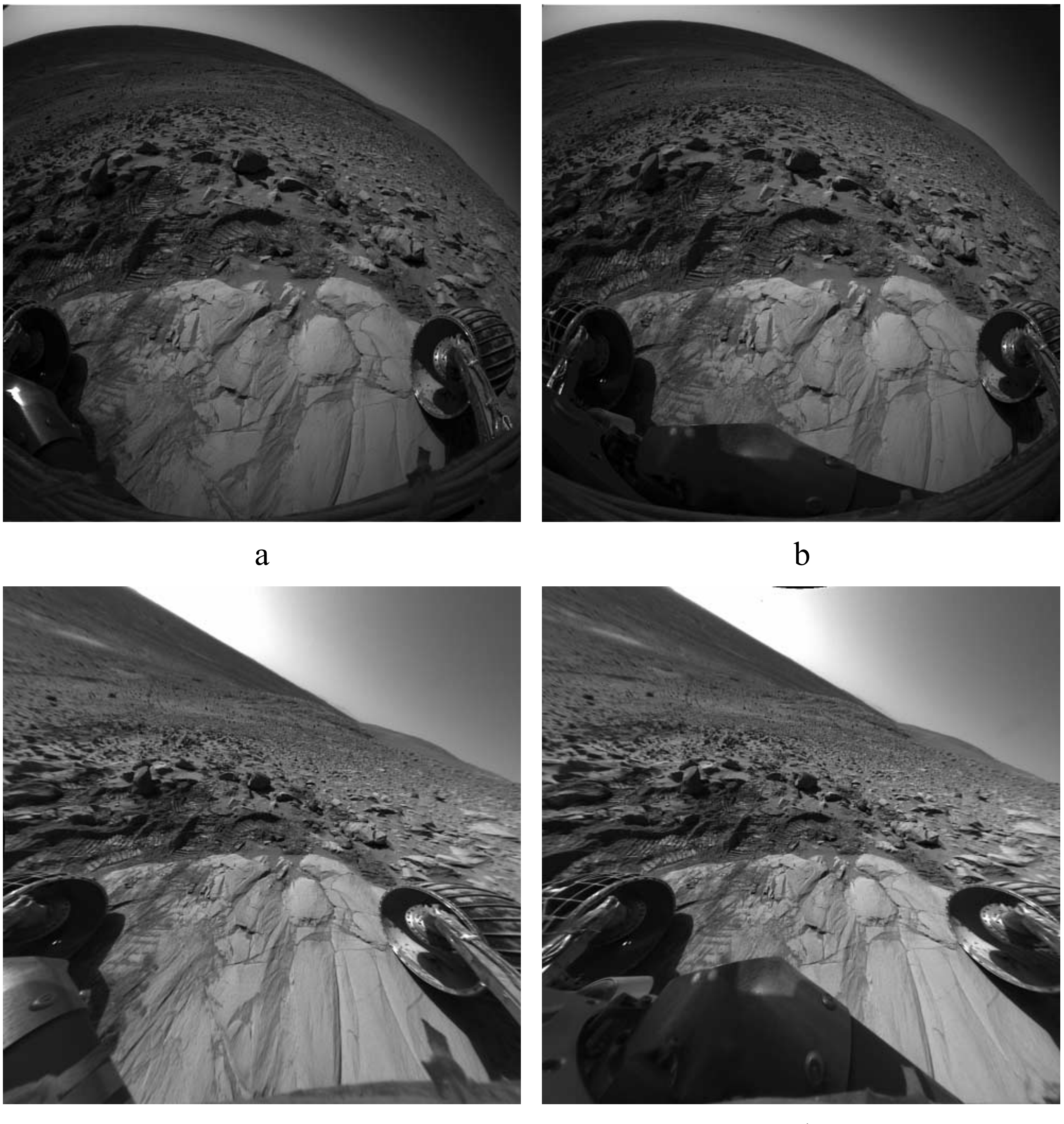


Figure 3. Front hazcam images acquired by Spirit on Sol 210 (Site 85). Figures 3a and 3b are the raw images (Right and Left, respectively, for cross-eyed stereo viewing). Figures 3c and 3d are the same images after linearization and radiometric correction (again, R and L for cross-eyed viewing).

was already 12-bit, this was just a copy. For consistency, this image was used as the base image for all further processing.

[45] Radiometric Correction (RAD, RAL, RSD, RSL, MRD, MRL): The images were radiometrically corrected by removing the effects of exposure time and temperature, and applying a flat-field correction [*Maki et al.*, 2003]. The technique used was called MIPLRAD. It is important to note that this was a quick-look radiometric correction intended mainly for operations work; full science-quality radiometric correction was performed by the pancam team [*Bell et al.*, 2003].

[46] Linearization (FFL, DNL, SFL): Each image was described by a camera model, which permitted a point in XYZ space to be traced into the image plane, and vice versa. The navcam, pancam, and MI used the CAHVOR model [*Gennery*, 2001], which modeled radial optical distortion. The front and rear hazcams used the more general

CAHVORE model [*Gennery*, 2002], which added a moving entrance pupil and could model fisheye lenses.

[47] The CAHVOR and CAHVORE models were generalizations of a much simpler linear camera model known as CAHV [*Yakimovsky and Cunningham*, 1978]. This consisted of four vectors representing the position of the camera entrance pupil (**C**), a unit vector normal to the image plane (**A**), and two vectors compositing the orientation of the CCD, the scale, and the image center in the horizontal and vertical directions (**H**, **V**). If **P** was a point in the scene, then the corresponding image locations x and y could be computed from

$$x = \frac{(\mathbf{P} - \mathbf{C}) \cdot \mathbf{H}}{(\mathbf{P} - \mathbf{C}) \cdot \mathbf{A}} \qquad y = \frac{(\mathbf{P} - \mathbf{C}) \cdot \mathbf{V}}{(\mathbf{P} - \mathbf{C}) \cdot \mathbf{A}} \tag{1}$$

The linearization process took the original image and reprojected, or warped, it so it could be described by the linear CAHV model. This process had several benefits:

[48] 1. It removed geometric distortions inherent in the camera instruments, with the result that straight lines in the scene were straight in the image.

[49] 2. It aligned the images for stereo viewing (epipolar alignment). Matching points were on the same image line in both left and right images, and both left and right models pointed in the same direction. With perfect calibration, line disparity would be 0.

[50] 3. It facilitated correlation, allowing the use of 1-dimensional correlators.

[51] 4. It greatly simplified the mathematics involved in using the camera model.

[52] However, the linearization process did introduce some artifacts: scale change and/or omitted data [*Deen*, 2003b]. It also introduced a small amount of interpolation noise (errors introduced by resampling the pixel grid, which is required when warping any image).

[53] The MER project decided early on to use linearized images for all operational terrain derivation. See Figures 3c and 3d for an example of linearized images.

4.2.2. Stereo-Image Products

[54] These products were derived from a pair of (linearized) stereo images. The design of the MER camera systems was such that stereo pairs were normally acquired simultaneously, an aspect used by the MIPL pipeline to match the pairs. Occasionally stereo pairs were acquired at different times; this required manual intervention to match the pairs.

[55] Disparity Map (DIL): Stereo image pairs were correlated to create a disparity map, which mapped each pixel in the left image to its matching pixel in the right image (to subpixel accuracy). This was the first, most important, and most CPU-intensive step in recovering 3-D terrain information.

[56] The correlation process was quite complex and is fully described by *Deen and Lorre* [2005]. In summary, downsampled (zoomed-out) linearized pairs were first correlated using the same 1-D correlator as was used by the onboard navigation software [*Goldberg et al.*, 2002]. This seed point image was then refined using a 2-D correlator, scaling up by a factor of 2 and repeating until full resolution was reached.

[57] An important variant on disparity was the process used to create photometric products by the pancam team [*Soderblom et al.*, 2004; *Johnson et al.*, 2006]. In this case, the original images were used: they were not linearized. Instead of a 1-D correlator, seeds were created using an idealized surface and the camera geometry. From that point on, the same 2-D correlator was used.

[58] XYZ Image (XYL): The XYZ image contained, for each pixel, the coordinates in 3-D space of the object imaged by that pixel. See Figure 4a for an example. This XYZ location was derived from the camera models and disparity map. For each pixel in the left eye, the disparity map was used to find its partner in the right eye. Rays were then projected into space from those pixels using the images' camera models. The point where these rays come closest to intersecting (midway between them) was the XYZ coordinate, which was stored in the file. This directly represented the 3-D terrain. The coordinate could be rejected for any of a number of reasons [*Deen and Lorre*, 2005]. These heuristically determined rules (e.g., diverging rays, line disparities or ray miss distances that are too large) helped to filter out points that were improperly correlated, which improved the quality of the result.

[59] Mask Image (MSL): Associated with each XYZ file was a mask, which was intended to exclude the rover volume and the horizon from consideration when making a mesh. This mask was not archived but was stored in the OSS.

[60] Range Image (RNL): The range image was a simple Cartesian distance of XYZ points from the camera center. See Figure 4b for an example.

[61] Terrain Wedge (VIL): This was a representation of the terrain for this image pair made by decomposing the XYZs into geometry triangles [*Wright et al.*, 2005].

[62] Surface Normal (UVL): This image contained the surface normal at each pixel. See Figure 4c for an example. This was computed by fitting a plane to all points from the XYZ image within 2.5 cm of the pixel, then rejecting outliers and repeating the plane fit. This process iterated until a minimum plane fit error was reached or until there were too few points remaining for a reliable plane calculation. The 2.5 cm threshold approximated the radii of the contact surfaces of the IDD instruments. The surface normal thus provided reachability and safety information for arm operations [*Leger et al.*, 2005].

[63] IDD Reachability (IDL): XYZ and Surface Normal were combined with Instrument Deployment Device (IDD) robot arm kinematics (from the flight software) to determine which points were reachable by the arm's four instruments. See Figure 4d for an example. Each instrument could be in one of four arm configurations (elbow up or down, wrist up or down) for a total of 16 reachability values for each pixel. For the Rock Abrasion Tool (RAT), instead of a simple flag, the value represented the maximum preload that could be applied by the arm at that point [*Leger et al.*, 2005].

[64] Surface Roughness (RUL): The roughness map was used to assess safety for the RAT. It contained for each pixel the maximum peak-to-peak deviation of

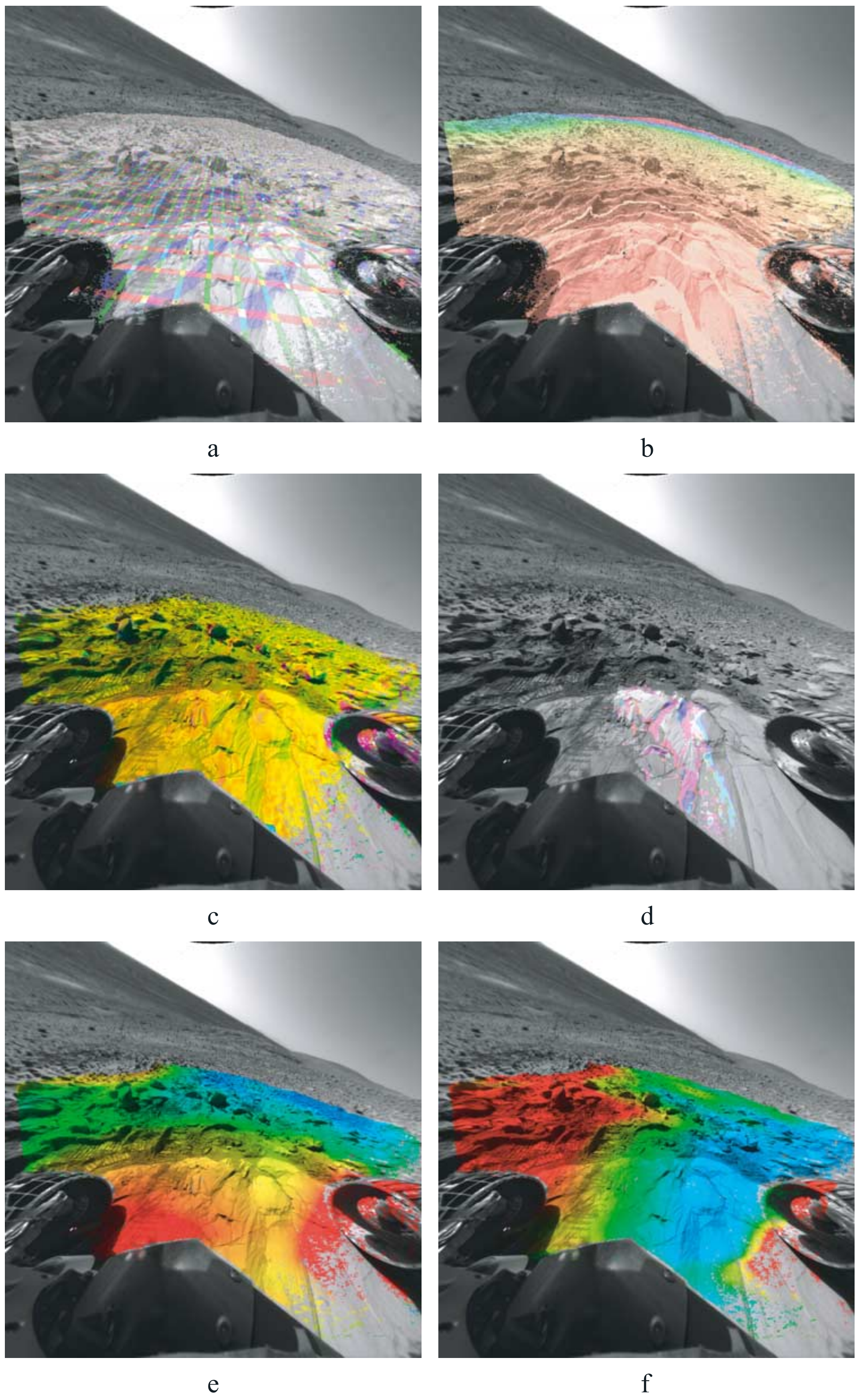


Figure 4. Derived images overlaid on hazcam from Figure 3. (a) XYZ image. Contour lines are 0.1 m apart; red for X, green for Y, blue for Z. (b) Range image. Contour lines are 0.1 m apart. (c) Surface normal. Color indicates direction of normal. (d) IDD reachability. Red is MI, green is MB, blue is APXS for one configuration. (e) Slope map. Blue is flatter, red is steeper (~30 degrees). (f) Solar energy map. Blue represents more energy; red represents less.

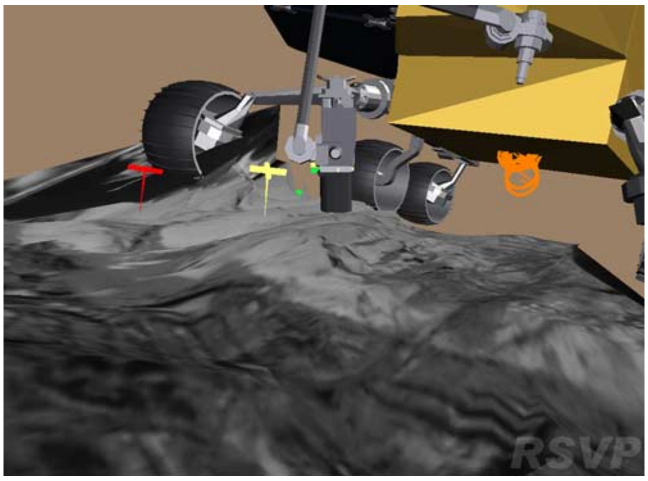


Figure 5. Screen shot of RSVP showing terrain mesh being used for arm (APXS) targeting. Mesh is derived from same front hazcam as Figures 3 and 4.

the surface from the local plane within an area the size of the RAT.

[65] Slope Products (SLL, SHL, SML, SRL): Late in the 90-Sol primary mission, as Opportunity descended into Endurance Crater and Spirit climbed the Columbia Hills, it was found that some extra products were needed to support operations. These were collectively called the Slope products, as well as Solar Energy. See Figure 4e for a Slope example. All of them were derived from the XYZ image and the Surface Normal. However, the surface normal is calculated somewhat differently from that described above, and was not saved or archived. Instead of a 2.5 cm patch, the normal was calculated over a rover-sized patch (1.6 m diameter). There were also a few other changes for efficiency over such a large patch.

[66] From the surface normal (whose three components are *u*, *v*, and *w*, corresponding to the components of the unit vector in the *x*, *y*, and *z* directions, respectively), the Slope (SLL) in degrees was computed as

$$\text{slope} = \frac{180}{\pi}\left(\frac{\pi}{2} + \tan^{-1}\left(\frac{w}{\sqrt{u^2+v^2}}\right)\right) \tag{2}$$

Slope Magnitude (SML) was directly related to slope, and is the magnitude of the normal unit vector projected into the horizontal plane:

$$\text{slope\_mag} = \sqrt{u^2+v^2} \tag{3}$$

Slope Heading (SHL) indicated the direction of the slope as a clockwise angle from north (in degrees, using the 4-quadrant form of arctangent):

$$\text{slope\_heading} = \frac{180}{\pi}\tan^{-1}\left(\frac{v}{u}\right) \tag{4}$$

Slope in the Rover Direction (SRL) was a measure of the component of the slope that was facing the rover, i.e., if the rover went radially outward from its current position, would that be a climb or a descent? It was intended to help optimize the usage of Spirit's right front wheel.

$$\text{srd} = -\frac{180}{\pi}\tan^{-1}\left(\frac{\mathbf{V}_x u + \mathbf{V}_y v}{-w}\right) \tag{5}$$

where **V** is a 2-D unit vector from the rover's position (**R**) to the location of the pixel (*x*, *y*, *z*):

$$\mathbf{V} = \frac{\left[x-\mathbf{R}_x,\ y-\mathbf{R}_y\right]}{\sqrt{\left(x-\mathbf{R}_x\right)^2+\left(y-\mathbf{R}_y\right)^2}} \tag{6}$$

[67] Solar Energy Maps (SEL): In addition to slope, the amount of available solar energy was a primary concern, especially through the Martian winter. This was computed by taking the dot product of the rover's normal with the approximate sun vector at noon. The sun vector was computed simply by using the sun angle at noon (*S*), which was changed on the order of monthly through the mission. Thus the Solar Energy map represented only an approximate insolation amount, but was good enough to avoid low energy areas during operations. Note that only the overall tilt of the rover was considered for this. Shadow effects were ignored; experience showed that very few local terrain features (such as rocks) were big enough to cast any significant shadow on the rover's solar panels. Self-shadowing, e.g., from the high-gain antenna and camera mast, was dealt with independently by the operations team. See Figure 4f for an example.

$$\text{energy} = \left[\cos\left(S\frac{\pi}{180}\right),\ 0, -\cos\left((90-S)\frac{\pi}{180}\right)\right]\cdot[u,\ v,\ w] \tag{7}$$

4.2.3. Multi-image Products

[68] Terrain Mesh Products (ASL): Terrain meshes were used by Rover planners to plan their traverses and IDD operations [*Wright et al.*, 2005]. A terrain mesh was derived from XYZ images and corresponding images that were used as the mesh "skin", or texture map. The mesh process began by taking the XYZ coordinates from the XYZ image and converting them to geometric triangles, a method called "triangulation". The point of each triangle was oriented toward the left eye/camera. Connectivity was implied by the pixel ordering or by volume-based surface extraction. This process was performed for each image to derive an associated terrain "wedge". Mesh building was completed by combining the many terrain wedges, thereby creating a unified 3-D terrain model for the rover's surroundings. See Figure 5 for an example.

[69] The mesh thus took the XYZ points and connected them into a polygonal representation of the surface suitable for use by 3-D modeling tools, with the image serving as a texture map. In this way the mesh could be viewed as a surface reconstruction of the ground near the instrument, capturing both the shape and visual features of the surface.

[70] Mosaic Products (CYL, PER, CYP, POL, VRT): Mosaics were composed of many images, combined to show the terrain from a wider perspective. The pipeline was capable of automatically generating mosaics, which was often done but not systematically. However, most

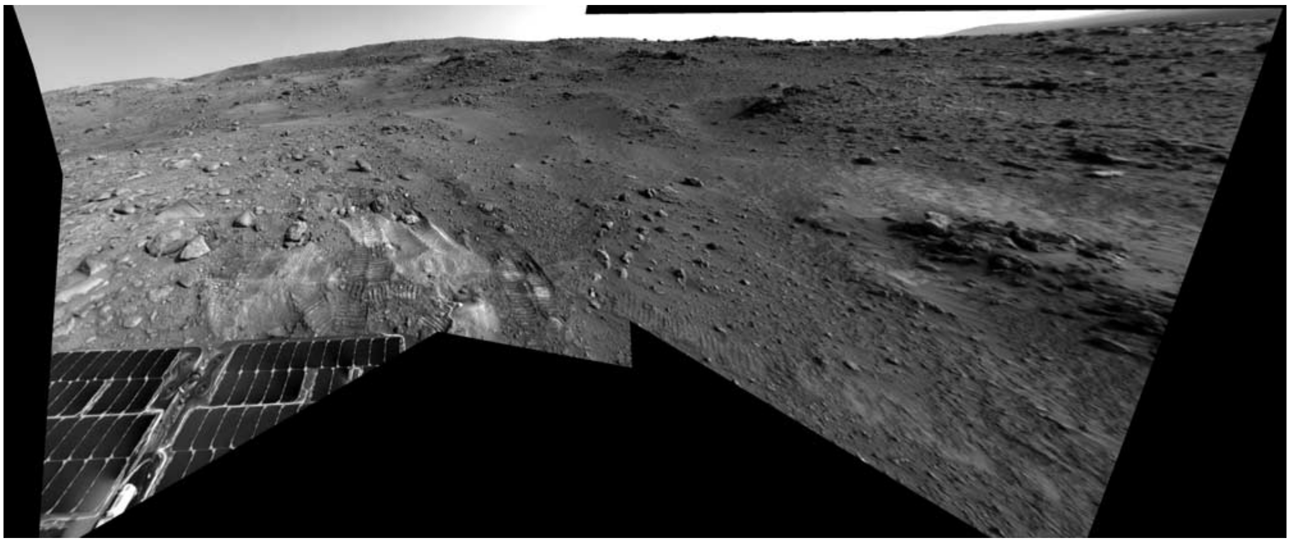


Figure 6. Perspective projection mosaic of part of Site 108 Position 139, from Spirit on Sol 424.

mosaics required manual correction to look their best. Mosaics and the correction techniques used for them are described in the next section.

4.2.4. Special Products

[71] Throughout the mission, MIPL was asked by various customers to create special products that were outside the normal automated processing flow. While each product was by definition different, a few examples are presented here to give a sense of the range of special products produced.

[72] Quite a few products were made for operations support. These included semi-standard tasks like changing the rover mask for a mesh or brightening up shadowed areas, to more specialized processing such as combining data from different locations into a single mesh. In several instances, the RAT team requested surface roughness maps generated using something other than the default parameter set. On occasion, high-priority imagery would not correlate well enough using the standard parameters, so parameter changes were needed. Slope products originated as a special product but soon became a standard product integral to mission planning.

[73] For science, by far the largest special request was for photometric products. This was a collaboration with the pancam team to generate photometric products systematically from a combination of MIPL and pancam software [*Soderblom et al.*, 2004; *Johnson et al.*, 2006]. Additional examples of science requests include enhancement of the sky to find clouds, analysis to look for dust devils, detailed (low-noise) XYZ processing of a trench dug by the wheels, and XYZ mosaics for coregistration with the miniTES spectrometer data.

[74] For PIO, examples included generation of short movies (animated GIFs) showing arm or rover motion, products combining MI images with hazcam, navcam, or pancam to provide context for the MI image, and mosaics of all varieties.

5. Mosaics

[75] The most visible products generated by MIPL for MER were mosaics. Many hundreds of mosaics were created for science, operations, and public outreach purposes. The mosaics created for MER fell into several broad categories.

[76] Full pancam panoramas: Created in collaboration with the pancam team, these 180° to 360° color and stereo mosaics were the signature mosaic products of the mission. As of this writing, there have been eight of these acquired by Spirit, named: Mission Success, Legacy, Bonneville, Santa Anita, Cahokia, Thanksgiving, Lookout and Independence. Opportunity has taken six, named: Mission Success, Lion King, Endurance 1 (180°), Endurance 2, Burns Cliff (180°) and Rub al Khali. MIPL performed the pointing correction and actual mosaicking of the images, using radiometrically corrected images produced by the pancam team. The completed mosaics were then handed off to the pancam team for final color processing. They were usually in cylindrical and cylindrical-perspective projections.

[77] Site panoramas: Each time the rover declared a new Site, or after a long drive, a navcam stereo panorama of the surroundings was taken. This was usually a full 360° panorama, although not always. MIPL systematically created mosaics for all of these, in all projections (except perspective). See Figures 6, 7, 8 and 9, which are different projections of the same Site panorama, for examples. They were used by long-term mission planners as well as providing the bulk of mosaics released on the public Web site.

[78] RAT holes: At the request of the RAT team, MI images covering every RAT hole were created in the perspective projection. A few of these were created in stereo using one of two methods: (1) physical parallax via arm motion and (2) depth reconstruction from focus [*Herkenhoff et al.*, 2006].

[79] MI mosaics: A large subset of MI targets were also mosaicked, most at the request of the MI team. These used the perspective projection.

[80] Special requests: A large number of ad hoc mosaics were created by special request from science team members, long-term planners, Public Outreach, and other sources. Many of these ended up in press conferences or Web site releases. These mosaics used a wide variety of projections and techniques.

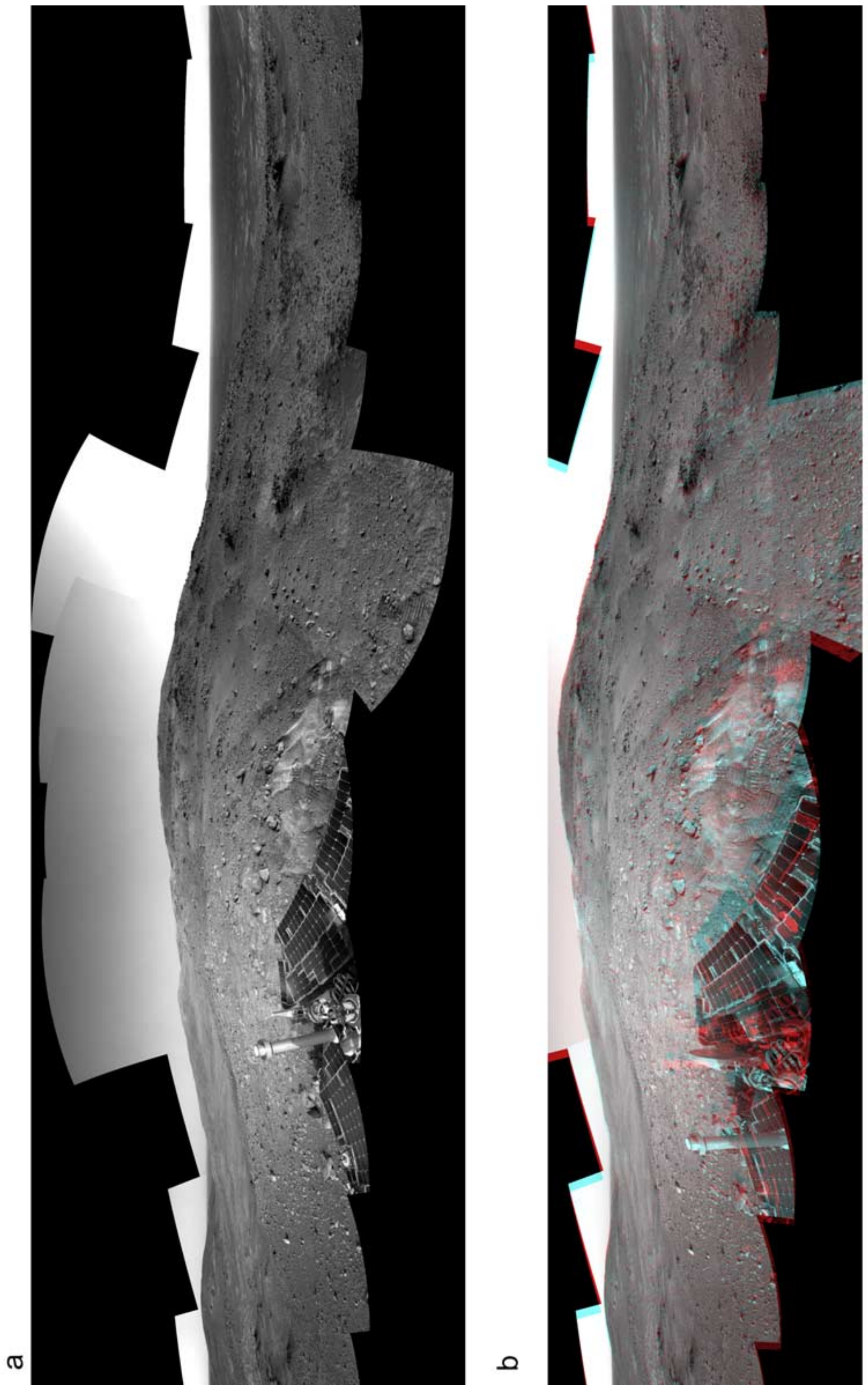


Figure 7. (a) Cylindrical projection mosaic of Site 108 Position 139, from Spirit on Sol 424. Hill is Husband Hill, with Tennessee Valley to the left. North is at left and right edges; south is in center. (b) Cylindrical-perspective stereo mosaic of the same scene. Use red/blue anaglyph glasses to view. Mosaic has been "untilted" to correct for rover tilt.

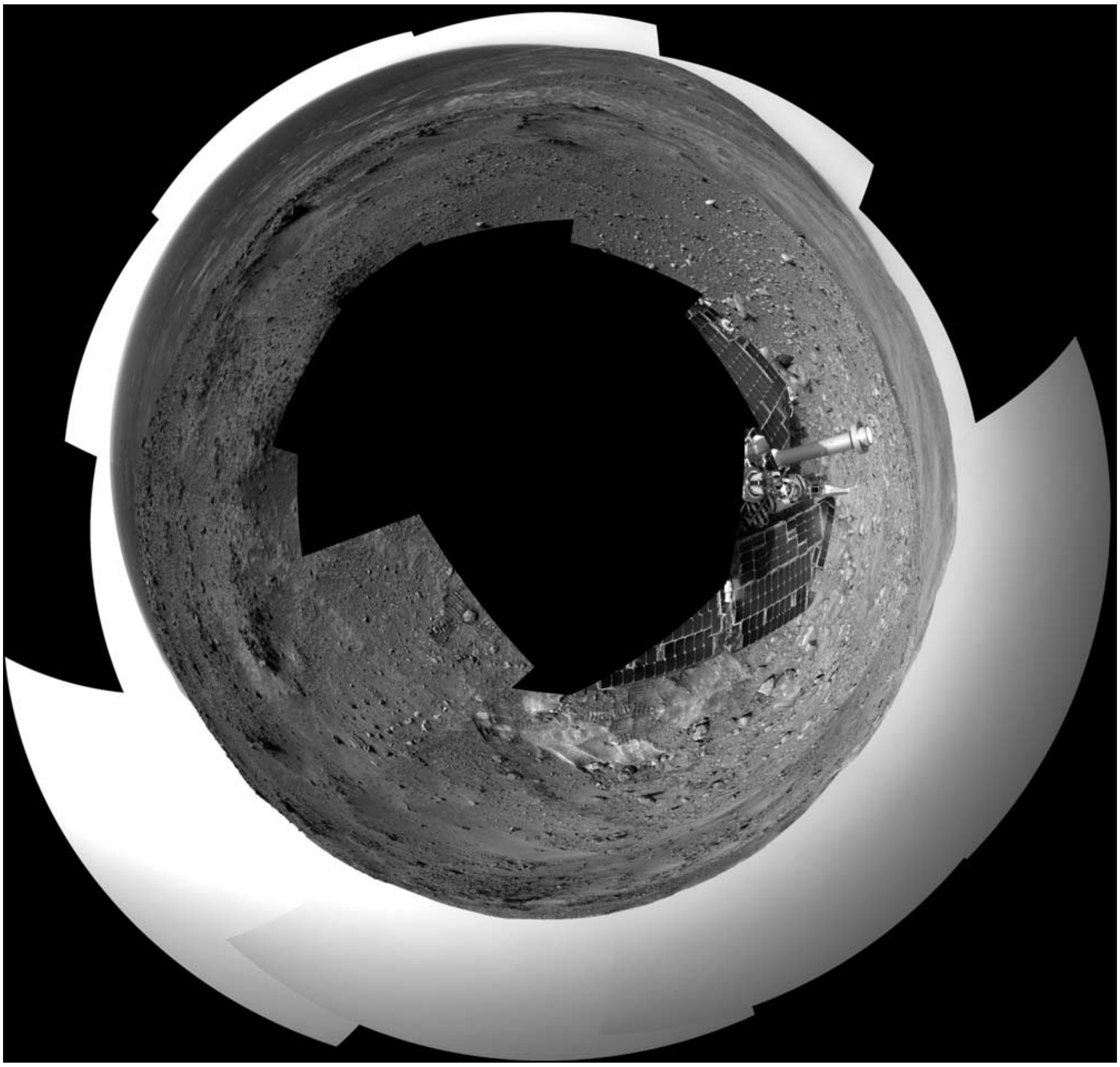


Figure 8. Polar projection mosaic of the same scene. North is up.

[81] In addition to the traditional mosaics of images, mosaics were also made from other RDR types. The most useful of these were mosaics constructed solely from XYZ, slope or solar energy RDRs. A coregistered image mosaic could be made using the same geometry, making them viewable via an overlay mechanism.

[82] The fundamental mosaic methodology and projections were substantially similar to those used for MPF [*LaVoie et al.*, 1999], although the software was largely rewritten and the metadata updated. The "untilt" feature (described below) of the cylindrical-perspective projection (called "McAuley" by LaVoie et al.) was new for the MER application, as were brightness correction and nonimage mosaics. The pointing correction process had its roots in the MPF process but has since been significantly updated.

5.1. Methodology

[83] This section describes the process used by MIPL to create mosaics. Conceptually, one can think of the process as adjusting the inputs, projecting them down to a surface, and looking at the result from a different point of view (the output projection). In reality, the process was run in reverse for ease of interpolation, as described below.

[84] The first step was to apply any pointing adjustments to the input images (see below). This resulted in an adjusted camera model that described the geometry of each input image. Note that pointing adjustment was optional; the telemetered camera models could have been used instead. This did not work well on Spirit due to some inaccuracies in the camera models; it worked better on Opportunity. Nevertheless, most mosaics destined for public release were corrected.

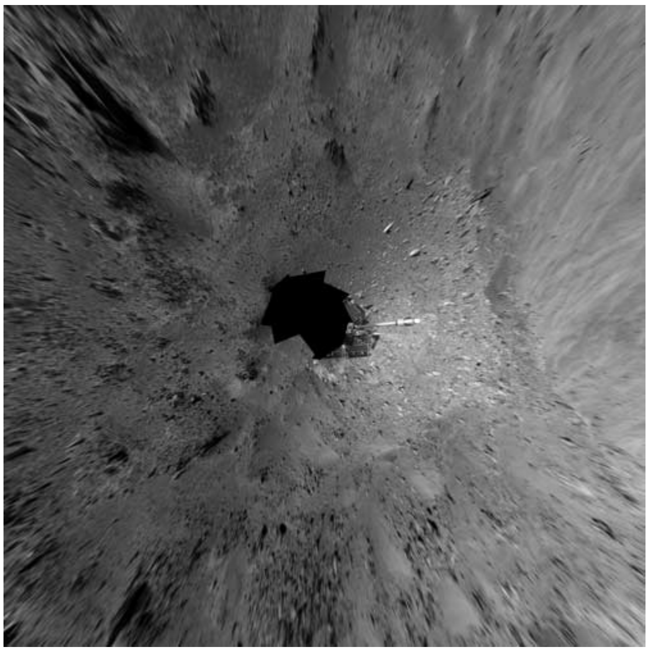


Figure 9. Vertical projection mosaic of the same scene. North is up, extent is ±15 m from center.

[85] Next, the output projection was determined. The projections are described in detail in the next section. The parameters describing the projection were stored in the image label [see *Alexander et al.*, 2003].

[86] A surface model was critical for mosaics. This was a mathematical surface which hopefully matched the actual scene. To the extent that the scene differed from the surface model, distortion, parallax, and uncorrectable seams could result. If the surface model matched the scene, there would be no parallax when changing the point of view (which is required when making mosaics). However, there were always variations, which introduced some amount of parallax in every mosaic. This was most evident on the solar panels where large seams were evident, while the ground features behind the panels matched perfectly.

[87] The most common surface model, used in all but a handful of mosaics, was a flat plane. This defaulted to a horizontal plane (normal pointing vertical) at the level of the nominal wheels. However, the pointing correction process usually determined an optimal surface plane, which was often slightly tilted and moved to compensate for local topography (it was a compromise over the entire area). The surface model for MI mosaics was especially problematic; the plane was highly tilted and proper placement was critical. Micro-topography often seen in (non-RATed) MI mosaics created quite severe parallax in some cases, which was why MI mosaics typically had much more visible seam mismatches than other kinds of mosaics.

[88] In addition to the plane, a sphere model was occasionally used for craters. A model projecting to infinity was also available, but as of this writing has not yet been used.

[89] After the surface model was constructed, the projection began. For each pixel in the output mosaic, a view ray in 3-D space was constructed. How this view ray was constructed depended on the projection type, and is detailed in the projection descriptions in the next section.

[90] The view ray was then projected out until it intersected with the surface model. The resulting point in XYZ space was used in the next step. If the ray did not intersect the surface, the point was assumed to be at infinity in the direction the view ray was pointing. An exception to that occurred for the Vertical projection, described below.

[91] The XYZ location (or direction for the infinity case) was then back-projected into each input image in turn, using the corresponding input camera model. The first input for which the resulting pixel coordinate was inside the image (excluding border pixels which were thrown away) stopped the process; that was the image from which the output pixel value was taken.

[92] Note that this had the effect of stacking the images such that the first one in the input list of images "wins". This was no feathering of overlaps; the first image was "on top" of all the others, and an image completely covered by preceding images was not used at all.

[93] The absence of feathering (combining or blending the images in an overlap area) was quite intentional. While careful blending could make the seam appear to vanish, it also introduced distortion and artifacts into the mosaic, which could adversely affect scientific interpretation. As for strict stacking, an alternative was to split the seam down the middle of the overlap. This was not done for two reasons. First, strict stacking led to a more predictable seam location, which had advantages in the pointing correction process. Second, stacking allowed the order to be adjusted. This permitted adjustments to the mosaic such as hiding bad shadows or data dropouts, or moving the seam from one edge to the other to avoid a problematic parallax issue.

[94] Finally, after the back-projection, a bilinear interpolation was performed on the input image, based on the four pixels surrounding the back-projected location. The result of this interpolation was the value of the output pixel. Brightness correction, if used, was applied just before the interpolation.

[95] Interpolation was optional, and was not normally done when making mosaics of XYZ or surface normal (UVW) data.

5.2. Projections

[96] MIPL supported five mosaic projections for MER: Cylindrical, Perspective, Cylindrical-Perspective, Polar, and Vertical. They all had different uses and are described below. In the equations below, which describe how the view ray was constructed, (i, j) represents the location of the output pixel in 0-based coordinates, with i corresponding to sample (x) and j to line (y) (the origin is in the upper-left corner). CAPITALIZED_VALUES represent projection parameters, which were stored by the same name in the PDS image label.

5.2.1. Cylindrical

[97] The cylindrical projection created a linear relationship between azimuth/elevation of the scene and line/sample in the image. Thus each pixel covered a constant number of degrees, which was the same in both directions. The

azimuth and elevation were measured from a single point, called the projection origin.

[98] Cylindrical projections were the primary panoramic mosaic. See Figure 7a for an example. They were unsuitable for stereo mosaics, however.

[99] Most cylindrical mosaics were produced using the Site frame, which removed the effect of rover tilt, resulting in a flat horizon.

[100] The view ray emanated from PROJECTION_ORIGIN_VECTOR at an azimuth and elevation defined as

$$\begin{aligned} \text{azimuth} &= \frac{i}{\text{MAP\_RESOLUTION}} + \text{START\_AZIMUTH} \\ \text{elevation} &= \frac{\text{ZERO\_ELEVATION\_LINE} - j}{\text{MAP\_RESOLUTION}} \end{aligned} \tag{8}$$

5.2.2. Perspective

[101] This mosaic was constructed by creating a synthetic output camera model, which was similar to the input camera models (except it was always linear, or CAHV). Thus the output looked very similar to the inputs. A perspective mosaic was only good for small areas however; as the field of view approached 180° the size of the mosaic approached infinity. 120° was about the maximum practical size for a perspective mosaic.

[102] Perspective mosaics were good for small-scale stereo mosaics (see Figure 6 for an example). MI mosaics exclusively used the perspective projection (see Figure 10 for an example).

[103] The view ray was constructed simply by projecting through the output camera model, which was stored in the label. The projection math is defined in equation (1).

5.2.3. Cylindrical-Perspective

[104] This mosaic, known as the "McAuley" [*LaVoie et al.*, 1999] during MPF, was a hybrid of cylindrical and perspective. It was linear in azimuth as with the cylindrical projection, but each column was assigned its own output camera model, which resulted in a perspective projection in the vertical direction. This resulted in a mosaic that was particularly well suited to stereo panoramic mosaics. See Figure 7b for an example.

[105] There were four steps to derive the output models:

[106] 1. The initial camera model was computed. This was a CAHV linearized model derived from the first input to the mosaic, and is described in the label.

[107] 2. The instantaneous field of view (ifov) of the "central" pixels (the point where the camera model **A** vector intersected the image plane) was computed:

$$\text{ifov} = \tan^{-1}\left(\frac{1.0}{|\mathbf{H} - \mathbf{A}(\mathbf{H}\cdot\mathbf{A})|}\right) \tag{9}$$

Alternatively, this could be derived from the image size and azimuthal extent (where the azimuths were adjusted by 360° such that the result was minimally positive):

$$\text{ifov} = \frac{\text{STOP\_AZIMUTH} - \text{START\_AZIMUTH}}{\text{LINE\_SAMPLES}} \tag{10}$$

[108] 3. The azimuth of each column was computed:

$$\text{azimuth} = \text{START\_AZIMUTH} + i \times \text{ifov} \tag{11}$$

[109] 4. The initial camera model was repointed using the camera kinematics routines (described below in the pointing correction section), using the computed azimuth and PROJECTION_ELEVATION. This resulted in the final camera model for the column.

[110] Finally, the view ray for each pixel was constructed as follows (**C**, **A**, **H**, **V** are the column's camera model parameters):

$$\begin{aligned} \text{samp} &= \mathbf{A}\cdot\mathbf{H} \\ \text{line} &= \mathbf{A}\cdot\mathbf{V} + j - \text{PROJECTION\_ELEVATION\_LINE} \end{aligned} \tag{12}$$

[111] This (samp,line) coordinate was then projected into space using the column's camera model, becoming the view ray.

[112] Note that the **C** points of the output cameras described a ring in space, whose diameter was approximately the baseline between the cameras, and whose plane was approximately horizontal in the rover frame. This ring maintained the baseline separation between the left and right eyes, and was what made this projection good for stereo panoramas.

[113] To achieve true, epipolar-aligned stereo, the mosaic had to be created using the Rover frame. This meant that the mosaic was aligned with the rover axes, but had the unfortunate effect of creating a sinusoidal horizon if the rover was tilted when the mosaic was acquired. If the tilt was small, the mosaic could be generated in Site frame, which resulted in a flat horizon but introduced some vertical disparity that made it hard to view in stereo. A recently developed technique mitigated this in some circumstances, resulting in a flat horizon with reduced (or ideally no) vertical disparity. This technique, called "untilting", pointed the individual cameras using a "flat" coordinate system for the kinematics computations rather than using rover frame. Thus the ring of **C** points was aligned with the horizon rather than the rover. In the absence of parallax, this worked well. However, the technique was rather sensitive to parallax effects caused by the scene not matching the surface model, so it was not always effective.

5.2.4. Polar

[114] The polar projection provided a panoramic view without the severe distortion near the nadir that cylindrical produces. This provided a better view close up but suffered somewhat at distance. See Figure 8 for an example. Radial distance from the center was linearly proportional to elevation, while azimuth described a circle at that distance.

[115] The view ray was computed by

$$\begin{aligned} x &= i - \text{SAMPLE\_PROJECTION\_OFFSET} \\ y &= \text{LINE\_PROJECTION\_OFFSET} - j \\ r &= \sqrt{x^2 + y^2} \\ \text{elevation} &= \frac{r}{\text{MAP\_RESOLUTION}} - 90^\circ \\ \text{azimuth} &= \text{REFERENCE\_AZIMUTH} + \frac{90^\circ - \tan^{-1}(y/x)}{\text{MAP\_RESOLUTION}} \end{aligned} \tag{13}$$

where the inverse tangent works for all four quadrants. The view ray emanates from PROJECTION_ORIGIN_VECTOR at the computed azimuth and elevation.

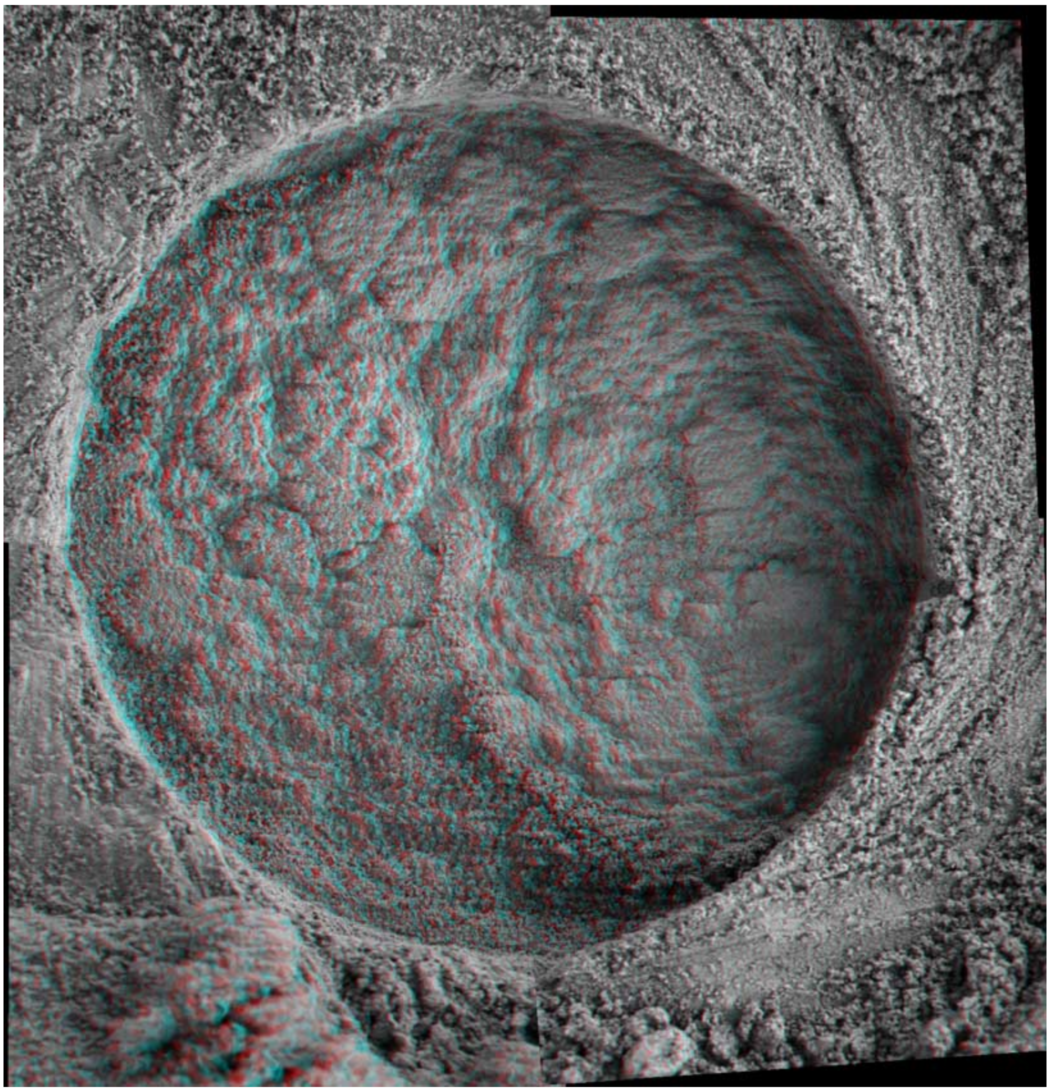


Figure 10. Perspective projection stereo mosaic of RAT hole "Diamond Jenness" taken by Opportunity on Sol 178. Stereo derived from focus by MI team.

5.2.5. Vertical

[116] The vertical projection provided a view where the location of a pixel in the image was directly proportional to the (x, y) coordinates of the projection surface. However, it is important to realize that this was not an orthorectified rendering. Nor was the (x, y) coordinate derived from actual stereo correlation. Instead, it assumed an ideal surface described by the surface model and simply projected to that. This caused severe layover of rocks and distortion of terrain that did not match the surface model. Nevertheless, it did provide a useful overhead view for many purposes. See Figure 9 for an example.

[117] The view ray was calculated as follows. Note that n_l is the number of lines in the mosaic and n_s is the number of samples.

$$\begin{aligned} x &= \left(\frac{n_l}{2} - j\right) \times \mathrm{MAP\_SCALE} \\ y &= \left(i - \frac{n_s}{2}\right) \times \mathrm{MAP\_SCALE} \end{aligned} \tag{14}$$

[118] The view ray emanates from $(x, y, 0)$ and points straight down $(0, 0, 1)$. If the ray did not intersect with the surface, it was changed to point straight up $(0, 0, -1)$ instead.

5.3. Pointing Correction

[119] The pointing of the cameras as provided in the rover data telemetry was determined to be sufficient for quick-look mosaics. However, there were errors in this pointing, both systematic and random. To achieve a high-quality mosaic, the pointing had to be corrected to reduce or eliminate visible seams. It is interesting to note that Opportunity did much better than Spirit in this regard; there was a $\sim 0.2°$ twist in both the pancam and navcam of Spirit that was not accounted for in the telemetered camera models.

[120] There were two methods to adjust the pointing: manual and tiepoint. Both are described below.

[121] Regardless of the method, the result of the pointing correction was a product called a "nav file". This file contained, for each image, the original pointing parameters

of the image, and the corrected parameters. It also contained additional identifying information.

[122] Pointing parameters were simply those numbers which represented how a camera was pointing in the rover coordinate frame, reduced to the available degrees of freedom. They modeled the physical actuators on the rover. Thus for the pancam and navcam, there were two pointing parameters: azimuth and elevation angles. The MI had four, one angle for each arm joint. The hazcams had none, since they were not articulated.

[123] The pointing parameters were used as inputs to kinematics procedures, which computed the camera model. These procedures modeled the physical motions of the cameras and transformed a calibration camera model to reflect the actual pointing and location of the camera. These kinematics procedures were used onboard to compute the camera models that were telemetered with the image. MIPL used the same procedures (even using some of the flight software) to recompute the camera models on the ground, using the corrected pointing parameters from the nav file.

[124] For the pancam and navcam, the MIPL code added one additional pointing parameter: twist. This was to compensate for the unmodeled twist described above for Spirit (although it was useful for Opportunity as well). This twist was implemented by rotating the final camera model by the specified amount around its **A** axis (i.e., the general direction in which the camera was pointing).

[125] Pointing correction therefore consisted of finding a set of pointing parameters for each image that minimized the seams. It is important to note that rubber-sheeting methods were not used. While these techniques could have completely eliminated seams, they compromised any scientific or operational use of the images. By maintaining each image as a fixed projection through its camera model, quantitative measurements could be obtained.

5.3.1. Manual Correction

[126] Manual correction was accomplished using a tool called MICA (Mosaic Interactive Correction Assistant). This was a Java-based tool that displayed the mosaic and allowed individual inputs to be selected and their pointing parameters interactively adjusted. The display repainted dynamically to show the result of the correction.

[127] This tool was used extensively early on in the mission. However, experience showed that the semi-automated tiepoint method was more efficient, and most mosaics were eventually created that way. The manual method was still used occasionally for adjusting the tiepointed results, however.

5.3.2. Tiepoint Correction

[128] The tiepoint method consisted of two phases: (1) gathering tiepoints and (2) generating a solution.

[129] Tiepointing was defined as finding coordinate pairs that identified the same feature from the scene in each of two different images. This resulted in pixels that matched, or tied, between the two images. Collecting a large number of these points across the mosaic helped to define how it was to be stitched together.

[130] An automated tiepointing procedure existed but did not perform well and was not used. Instead, tiepoints were gathered interactively by an analyst using a program called marstie to analyze the image geometry and display overlapping pairs one at a time on a monitor screen. For each overlapping pair, the user manually picked matching features on the two images. The program's correlator gathered the resulting tiepoints and processed them to subpixel accuracy. Most commonly, three tiepoints were gathered along a long seam (adjacent images horizontally or vertically), with one tiepoint in a corner overlap. But that was just a guideline; actual numbers varied.

[131] Because the images were adjusted on the basis of the tiepoints, they tended to be most accurate around the tiepoints. For this reason, tiepoints were gathered close to the edge of the image that was on top in the final mosaic. This put the tiepoints right next to the seam, which tended to minimize the seams. Tiepoints were also generally placed as close as practical to the nominal surface (so that they matched the surface model).

[132] It typically took an hour or less to tiepoint a navcam or MI mosaic. However, large pancam panoramas could take several days or longer to do.

[133] After the tiepoints were gathered, a program called marsnav analyzed them to come up with a solution. This was done using a function minimization process, where the pointing parameters for each image were the free parameters. The minimization algorithm was called "amoeba" and is defined by *Press et al.* [1988, p. 305ff]. It was a downhill simplex method that did not require partial derivatives to operate.

[134] The function being minimized was the residual tiepoint error. For each tiepoint, the coordinate from one image was projected from that image to the surface model and back into the other image, just as for a mosaic. The projected location was then compared with the second coordinate from the tiepoint. The difference in each direction (x and y) was squared, and the sum across all tiepoints was accumulated. This residual error was the cost function that was minimized. As the pointing parameters were adjusted, the residual error became smaller when the tiepoints matched better. Once the minimum was obtained, the pointing parameters were saved in the nav file.

[135] In addition to adjusting pointing parameters, the same function minimization procedure could also adjust the surface model, and even perform rover localization. It became a standard practice to let marsnav determine the appropriate surface model for the mosaic. Rover localization (changing the rover's position and/or orientation) was used on occasion, when images from multiple locations were combined, but this was rare. Both worked by simply adding more free parameters to the minimization process in addition to the pointing parameters; they affected the projection, which changed the cost metric without changing the cost function itself.

[136] A recent addition to this process was the concept of "inertia". As originally designed, the solution was insensitive to a global change to all images, say adding the same amount to all azimuths. This could cause the result to "walk", giving inaccurate azimuth results. In practice, this did not occur with full 360° mosaics, but occasionally was an issue for small ones. Another issue was the occasional misalignment of the horizon for small mosaics with frame twist (a la Spirit). Inertia added an additional factor to the cost function, which made the images tend to stay in place (hence the name). This factor was the sum across all pointing parameters of the difference between the original

and current pointing parameters, times a weight, squared. The weight determined how strong the inertia effect was. This proved to be a very valuable addition and was especially useful when making stereo mosaics (to preserve the original baseline as much as possible).

5.4. Brightness Correction

[137] Almost all MIPL mosaics used radiometrically corrected inputs: either MIPLRAD for navcam and MI mosaics, or data provided by the pancam team for pancam mosaics. However, these methods were not always perfect. In roughly 25% of the mosaics, differences in lighting created radiometric discontinuities at the seams.

[138] In order to reduce these discontinuities, a multiplicative factor was applied to each image. A pair of programs (marsint/marsbias) analyzed the overlaps and determined the optimum factors via another function minimization process. This eliminated most of the brightness seams, but not all of them. A process that varies the correction factor within each image, leaving only discontinuities caused by differences in illumination angle, is being considered for future work.

6. Rover Localization

[139] Each time the rover moved, camera image data were acquired from a different point of view. Often it was desirable to combine data taken from two different locations and combine them in some manner. This was especially useful for mosaics and terrain meshes. In order to do that, knowledge of the relative position and orientation of the rover at each location was required.

[140] Determining the position (XYZ location) and orientation (roll/pitch/yaw) of the rover (collectively called "location" here) was a process called localization. The baseline localization was provided by the rover itself in telemetry: it maintained a knowledge of where it was based on counting wheel rotations or performing visual odometry [*Olson et al.*, 2001], and of how it was oriented via tilt sensors and finding the sun with the cameras. Unfortunately, the rover's position knowledge was not very precise; it could be 10% or more off in some cases. Orientation knowledge was better but errors still occurred.

[141] Therefore it was often advantageous to refine the rover's location (or more commonly, just its position) on the ground via various techniques. Several teams generated localization updates throughout the mission, most notably the global position refinements from R. Li (see methodology of *Li et al.* [2004]). The Mobility operations team also generated refinements from time to time.

[142] MIPL used several methods for localization, which are described below.

6.1. Site and Rover Vector Files

[143] The rover used a Rover Motion Counter (RMC) to keep track of its location. This counter incremented every time the rover or any significant part of the rover (arm, mast, HGA) moved [*Maki et al.*, 2003]. Of interest here are the first two components of the RMC: Site and Position.

[144] Each new Site identified by the operations team defined a new coordinate system with the rover at (0, 0, 0), and zeroed out all the counters. Then the Position within that Site incremented each time the rover moved. All rover positions were relative to the Site coordinate frame. Incrementing the Site allowed for accumulated errors in rover position to be zeroed out, and made it easier to work in the local area.

[145] Each image acquired by the rover had associated with it the RMC counter value, and the rover's idea of where it was relative to the Site and its orientation (orientations were always defined with respect to north and the gravity vector). This information was stored in the image header (labels). It was also stored in a file called the Rover Vector File (RVF). These files (one per Site) existed in the OSS and were a convenient way to look at the rover motion history.

[146] Unfortunately, things were not so easy when it came to Site offsets. A Site cleared the coordinate system for local work, but the location of that Site origin relative to the previous Site was necessary to keep track of global position. This offset between Sites was not available in the image header. It came down in telemetry only in a product called the SAPP Knowledge Report. The information was captured by MIPL and stored in a file called the Site Vector File (SVF). There was only one of these per mission, and was the most convenient place to find all inter-Site vectors.

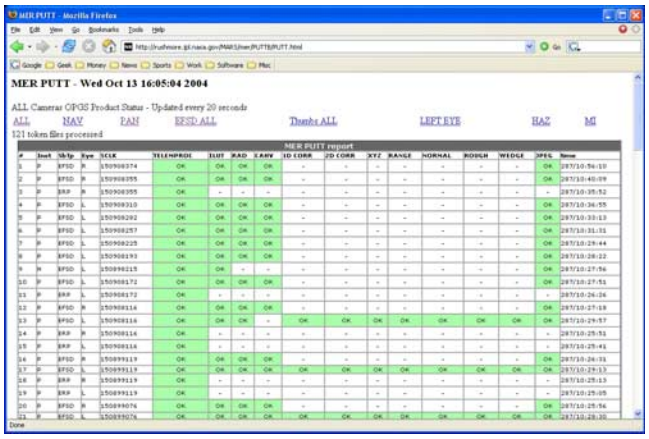
[147] The original design called for updates to the SVF and RVFs to be made on the basis of new localizations. However, operational procedures were not developed to handle these updates. The result was that the SVF and RVF files in the OSS contained only the telemetered rover locations. Localization updates were therefore maintained on an ad hoc basis via special SVF/RVF files by MIPL, or via other (often incompatible) means by other teams.

6.2. MIPL Localization Techniques

[148] The first method for rover localization attempted to coregister several pancam mosaics of Opportunity's Eagle Crater outcrop. First, a low-resolution navcam mosaic was created. The pancams were then mosaicked into the same projection on the basis of telemetered positions. The mosaics were compared to the low-res version and each other, and offsets were estimated by hand. The process then repeated using the new estimates. This required a massive amount of human effort, with results that were less than satisfying.

[149] The second method consisted of analyzing XYZ locations. The coordinates of matching features were extracted from images taken from different locations. The offset between the two was applied to the rover position to get an updated position, the XYZs were rerun, and the process was repeated until it converged adequately. This provided a much better alignment of features in the final mosaic.

[150] The difficulties of these methods motivated the development of a third technique after a couple months of operation. This technique included localization parameters (position and/or orientation for selected values of the RMC) in the function minimization of the marsnav program, and was described in more detail in the Pointing Correction section above. The optimal localization was thus computed at the same time as the mosaic pointing correction and


MER PUTT - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://rushmore.jpl.nasa.gov/MARS/mer/PUTTB/PUTT.html
Google Geek Money News Sports Work Software Misc
Done

MER PUTT - Wed Oct 13 16:05:04 2004

ALL Cameras OPGS Product Status - Updated every 20 seconds

ALL NAV PAN EFSD ALL Thumbs ALL LEFT EYE HAZ MI

121 token files processed

MER PUTT report

#	Inst	SbTp	Eye	SCLK	TELEMPROC	ILUT	RAD	CAHV	1D CORR	2D CORR	XYZ	RANGE	NORMAL	ROUGH	WEDGE	JPEG	Time
1	P	EFSD	R	150908374	OK	OK	OK	OK	-	-	-	-	-	-	-	OK	287/10:56:10
2	P	EFSD	R	150908355	OK	OK	OK	OK	-	-	-	-	-	-	-	OK	287/10:40:09
3	P	ERP	R	150908355	OK	-	-	-	-	-	-	-	-	-	-	-	287/10:35:52
4	P	EFSD	L	150908310	OK	OK	OK	OK	-	-	-	-	-	-	-	OK	287/10:36:55
5	P	EFSD	L	150908282	OK	OK	OK	OK	-	-	-	-	-	-	-	OK	287/10:33:13
6	P	EFSD	L	150908257	OK	OK	OK	OK	-	-	-	-	-	-	-	OK	287/10:31:31
7	P	EFSD	L	150908225	OK	OK	OK	OK	-	-	-	-	-	-	-	OK	287/10:29:44
8	P	EFSD	L	150908193	OK	OK	OK	OK	-	-	-	-	-	-	-	OK	287/10:28:22
9	N	EFSD	L	150898215	OK	OK	-	-	-	-	-	-	-	-	-	OK	287/10:27:56
10	P	EFSD	L	150908172	OK	OK	OK	OK	-	-	-	-	-	-	-	OK	287/10:27:51
11	P	ERP	L	150908172	OK	-	-	-	-	-	-	-	-	-	-	-	287/10:26:26
12	P	EFSD	R	150908116	OK	OK	OK	OK	-	-	-	-	-	-	-	OK	287/10:27:18
13	P	EFSD	L	150908116	OK	OK	OK	-	OK	OK	OK	OK	OK	OK	OK	OK	287/10:29:57
14	P	ERP	R	150908116	OK	-	-	-	-	-	-	-	-	-	-	-	287/10:25:51
15	P	ERP	L	150908116	OK	-	-	-	-	-	-	-	-	-	-	-	287/10:25:41
16	P	EFSD	R	150899119	OK	OK	OK	OK	-	-	-	-	-	-	-	OK	287/10:26:31
17	P	EFSD	L	150899119	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	287/10:29:13
18	P	ERP	R	150899119	OK	-	-	-	-	-	-	-	-	-	-	-	287/10:25:13
19	P	ERP	L	150899119	OK	-	-	-	-	-	-	-	-	-	-	-	287/10:25:05
20	P	EFSD	R	150899076	OK	OK	OK	OK	-	-	-	-	-	-	-	OK	287/10:25:56
21	P	EFSD	L	150899076	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	287/10:28:30

Figure 11. Screen shot of Process Update Tracking Tool (PUTT) display.

surface model determination. This technique was used a few times, most notably for the Spirit "Legacy" 360° color panorama, which was acquired from two slightly different locations.

[151] While this later technique worked reasonably well, it was still a highly labor-intensive process, and was not used very often. Combining data taken from different locations was inherently problematic; besides the localization issue, parallax was a huge issue for most geometries.

[152] MIPL also made use of localizations provided by other teams, most notably for several terrain meshes made using data from different locations with localizations provided by the Mobility team. This was most useful when the rover needed to backtrack into previously traversed terrain.

7. MIPL Operations

[153] The MIPL team was busiest during and immediately following the telemetered downlink of MER instrument data. Through these time periods, MIPL analysts had to keep close tabs on the pipeline processing of input data and the production of a wide variety of output products. Special requests varying in complexity and purpose were often submitted to the MIPL team by any number of product customers, including rover engineers, science team members and JPL Public Information Office (PIO) personnel. As a result, MIPL was involved in passing a large number of standard and special products to PIO for release to the public.

7.1. Product Tracking

[154] To address product accountability issues, MIPL analysts developed data tracking tools to monitor the data flow through the pipeline. The Unix environment available on the GDS enticed most tool development to be at the shell script level, though the need for low-level coding was not eliminated. This section discusses a few of MIPL's more noteworthy data product tracking capabilities.

7.1.1. Script-Based

[155] The MIPL pipeline's reliance upon directory queues in the OSS allowed a simple Perl script to be written that provided insight into the systematic processing. It was not a GUI representation, but numerical, counting file totals in each queue. This capability served as a primary view into the status of the pipeline's processing.

[156] Other script tools were written to determine which data products were completely processed and which ones were partial products. These tools were easy to develop due to the pipeline's flexible design, and were very useful for data accountability.

7.1.2. GUI-Based

[157] A GUI system called the "Process Update Tracking Tool" (PUTT) was created that presented the MIPL system operator with a Web page which visually indicated (via color) the completion status of each application program executed within the MIPL pipeline. An application program that completed successfully was marked in green, while a program that completed unsuccessfully was marked in red. From this information, the operator could (1) quickly determine if the pipeline was processing the data nominally, (2) isolate processing errors, and (3) interactively link to the appropriate section of the processing log files that included the processing error. See Figure 11 for a captured snapshot of the PUTT tool's GUI.

[158] The PUTT system was implemented using a system of token files. As the pipeline started processing each EDR, it wrote a small XML token file. This file was updated each time an application program was run with the program name, return status, time, and a pointer to the log file. To protect against simultaneous overwriting of token files by multiple processes, the PUTT system invoked a special program that write-locked the current file, so that it was updated by only one process at a time.

[159] These token files were collected every few minutes by a Perl script, which concatenated them together and transformed the result into an HTML file. If errors were present, a second page containing the log file extract was created as well. These files were then pushed to a Web server for viewing (which had to be outside the flight LAN, due to project constraints).

7.2. Product Quality Control

[160] In order to assure that the data products produced by the MIPL team were properly created, a number of quality control (QC) procedures were incorporated into the pipeline process. Quality control consisted of visually inspecting image data products and visually inspecting the label associated with the data products.

[161] The earliest possible QC step consisted of examining the input products to the MIPL Pipeline process. As described previously, data products came to MIPL as separate instrument data and metadata files. Team members used special GDS tools (mer_dp_view, showmerdpimg) to view these input products in order to verify whether visible data corruption was present prior to MIPL processing.

[162] For visual inspection of nearly all products created by the pipeline, (EDRs and RDRs), a Java program was written to aid in viewing of all products related to a particular EDR. This tool, marsviewer, allowed the analyst to overlay the various RDR products graphically on top of the original or linearized image to verify their correctness. This RDR overlay capability was used to create the various RDR images in Figure 4. Moving the cursor over the image showed pixel values of any or all of the related products, from intensity values to range, slope, XYZ coordinates, or

even IDD reachability values. In addition to its extensive use as a QC visual inspection tool, marsviewer quickly became a favorite application for data interpretation amongst members of the science and engineering teams.

[163] While marsviewer was used for visual QC of most products, the wedges, meshes and mosaics were inspected using other software. In the case of the wedges and the meshes, the products were inspected with the target selection programs SAP and RSVP. The mosaics were viewed with a more generic image display program.

[164] Because of the high volume of the data flowing through the pipeline and the timing requirements placed on the MIPL team, visual QC was performed daily on a spot-check basis for each downlink, and in response to reported problems. The MIPL analyst would browse the newly arrived products and verify that they looked "normal", but they would not necessarily view every product created by the pipeline.

7.3. External Delivery and Access

[165] There were a significant number of users external to the flight LAN, or even external to JPL, which required access to the data products generated by the pipeline. This access was provided via two primary mechanisms: FEI and the Image Atlas.

7.3.1. External Product Delivery

[166] Products generated by the MIPL pipeline were delivered from the OSS to remote sites external to the flight LAN using a client-server data distribution system developed at JPL called FEI [*Huang*, 2003]. MIPL administers FEI servers at JPL to manage mission data for a variety of flight projects.

[167] For MER, FEI client software installed on machines at sites located outside the flight LAN, such as the home institutions of science teams and other JPL local sites, transferred data to and from the FEI servers. FEI works with Kerberos, a security package developed through Project Athena at MIT to provide a means of authenticating users to ensure security over a network using encrypted passwords. By using Kerberos [*Neuman and Ts'o*, 1994], FEI maintains its own individualized user access control, so FEI users are not required to establish accounts on any data center machines to access data, which is the case with FTP. The strength of FEI is in its subscription capabilities. A subscription allows files that are added to an FEI server to be pushed automatically to subscribed sites. For MER, FEI subscriptions were used to parse EDR and RDR labels and add the meta-data to MIPL's internal database for the Image Atlas (see next section).

[168] FEI subscriptions were also used to support remote operations, beginning with the first MER Extended Mission. As science teams returned to their home institutions, remote copies of the OSS replicated at those institutions were populated by MIPL data products via FEI. This allowed remote use of SAP, greatly reducing Extended Mission travel costs.

7.3.2. PDS Access to Products

[169] The MIPL pipeline produced a huge quantity of products. In order to facilitate access to these, a Web-based PDS tool called the Image Atlas had the capability of querying a MIPL-resident database for quick retrieval and display of products. While not used for tactical operations, it was very valuable for finding and retrieving data for analysis.

[170] MIPL's database, located external to the flight LAN, was populated with the metadata information carried in each product's PDS label. The queries available via the Image Atlas were based on the designs implemented for MPF and other flight projects, providing the user with quick visibility to the data as well as a means to download it to local resources.

[171] The Image Atlas can be accessed at http://pdsimg.jpl.nasa.gov/Atlas.

[172] External to JPL at Washington University in St. Louis, the PDS Geosciences Node maintained a similar Web-based capability called the Analyst's Notebook for the science community to fetch MER science and engineering data and documentation.

[173] The Analyst Notebook can be accessed at http://anserver1.eprsl.wustl.edu.

7.4. Public Release of MIPL Products

[174] Camera image products generated by MIPL as part of OPGS were publicly released along three different timelines throughout the mission via a variety of outlets: (1) press conferences convened intermittently to release selected subsets of OPGS and science team products, (2) a Web-based browser to view JPEG compressed versions of image EDRs released daily to a nationwide alliance of museums, and (3) PDS Archive Volumes released every two or three months that contained OPGS and science products and ancillary documents produced to PDS standards.

[175] Regarding the press conferences, members of the MER flight team were encouraged to submit candidate images for press release. A process was defined that allowed MIPL analysts to designate products for consideration as press release images. Press briefing candidate images were provided in both TIF and JPEG format and supplemented with a text file containing a brief caption. Candidate images and their captions were placed in a designated directory location for consideration by a team of reviewers. Special video products created by the Solar System Visualization (SSV) team were also produced for press conferences, allowing the presenting scientist or engineer to walk the public and the press through the remote scene on Mars as it panned across the video screen.

[176] The second process for public release of MIPL products required MIPL to electronically deliver contrast enhanced JPEG versions of the most recent image EDRs to PIO each day. This allowed interested members of the public to browse the latest images from the surface of Mars soon after they were processed. These products were created and transferred to a server outside the mission firewall via FEI as part of the pipeline process.

[177] As the third outlet for public consumption of MIPL products, at scheduled intervals every two or three months subsequent to the MER prime mission, various nodes of the PDS assembled inventories of OPGS and science team products in 90-Sol increments into archive volumes built according to specific PDS standards. The primary means to access the released PDS data sets were the Image Atlas and Analyst's Notebook mentioned in section 7.3.2. References to the Spirit rover camera data sets that were produced by OPGS and archived into PDS volumes are too many in

number to list in this paper. They are available at http://starbrite.jpl.nasa.gov.

8. Conclusion

[178] The system developed by MIPL borrowed from the old and the new to forge itself into a highly successful asset for MER mission operations. It incorporated legacy capabilities from previous Mars lander missions in a successful bid to reduce development costs. In addition, it yielded new innovations evoked by the demands and time constraints of the in situ operations scenario. These new capabilities included a robust pipeline, many new types of RDR derived products, terrain meshes, nearly seamless mosaics and new rover localization techniques, all managed using a number of QC, tracking and data distribution tools. Capabilities demonstrated for the first time in support of MER were designed to be reusable, and will become the baseline for future planetary exploration missions, including the upcoming Phoenix and Mars Science Laboratory missions and the Lunar Exploration Program.

Notation

ASCII	American Standard Code for Information Interchange.
APXS	Alpha Proton X-ray Spectrometer.
CAHV	Linear camera model described by vectors C (**C**enter of focus), A (**A**xis normal to the sensor plane, which is not the optical axis), H (**H**orizontal information) and V (**V**ertical information).
CAHVOR	Camera model CAHV accounting for CCD and nonlinear distortions with vector O (**O**ptical axis) and coefficients R (**R**adial distortion).
CAHVORE	Camera model CAHVOR accounting for E (moving **E**ntrance pupil).
CCD	Charge Coupled Device.
EDR	Experiment Data Record, a NASA Level 0 product.
FEI	File Exchange Interface, a server/client based file delivery system.
FIDO	Field Integrated Design & Operations, a JPL rover testbed.
FTP	File Transfer Protocol, a protocol for exchanging files over the Internet.
GDS	Ground Data System.
GIF	Graphics Interchange Format, an image file format.
GUI	Graphical User Interface.
Hazcam	Hazard and Avoidance Camera, an engineering stereo camera.
HGA	High Gain Antenna.
HTML	Hypertext Markup Language.
IDD	Instrument Deployment Device.
JPEG	Joint Photographic Experts Group, a lossy image compression technique.
JPL	Jet Propulsion Laboratory.
LAN	Local Area Network, a firewall secured network.
M01	Mars 2001 lander testbed.
MB	Mössbauer, a spectrometer instrument.
MER	Mars Exploration Rover.
MI	Microscopic Imager.
MiniTES	MiniatureThermal Emission Spectrometer.
MIPL	Multimission Image Processing Laboratory, at JPL.
MIPLRAD	MIPL's radiometric correction process.
MPF	Mars Pathfinder.
MPL	Mars Polar Lander.
MSL	Mars Science Laboratory.
Navcam	Navigational Camera, engineering stereo camera with wide field of view.
NFS	Network File System.
OPGS	Operations Product Generation Subsystem.
OSS	Operations Storage Server.
Pancam	Panoramic Camera, science stereo camera with narrow field of view.
PDS	Planetary Data System.
PIG	Planetary Image Geometry, a software library of core subroutines.
PIO	Public Information Office, at JPL.
PUTT	Process Update Tracking Tool, a GUI-based quality assessment tool.
QC	Quality Control.
RAT	Rock Abrasion Tool.
RDR	Reduced Data Record, a NASA Level 1+ product.
RMC	Rover Motion Counter.
RSVP	Rover Science Visualization Planner, the primary navigation tool used by the Rover Planner team.
RVF	Rover Vector File.
SAP	Science Activity Planner, the primary target selection tool used by science teams.
Sol	Mars solar day.
SSV	Solar System Visualization, a team of visualization experts at JPL.
SVF	Site Vector File.
TIF	Tagged Image File, a grey-scale and color image file format.
VICAR	Video Information Communication And Retrieval, MIPL's image processing software.
XML	Extensible Markup Language.

[179] **Acknowledgments.** The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The authors would like to extend their great appreciation to Ken Herkenhoff and his Microscopic Imager Team, to Jim Bell and his Pancam Team, to Brian Hammer and his GDS Team, and to Steve Spohn and his MDOT Team for their camaraderie during the many months of testing and supporting MER landed operations. We also wish to thank Justin Maki, John Callas, Jim Erickson, Joy Crisp, Ray Arvidson, Steve Squyres, Eric DeJong, Larry Preheim, and Sue LaVoie for their guidance and inspiration throughout the MER mission. Particularly, we would like to acknowledge the following individuals who were involved with some facet of the data production system described in this paper: Hyun Lee, Elmain Martinez, Richard Norman, Steve Levoe, Nicholas Toole, Rafael Alanis, Betty Sword, Pam Woncik, Myche McAuley, Jim King, Greg Earle, Kris Capraro, Alan Runkle, Bill Stromberg, Maher Hanna, and Janet Fung. Finally, our special gratitude is offered to Jack Morrison, Costin Radulescu, and Robert Phipps for their tireless development efforts.

References

Alexander, D., H. Mortensen, and R. Deen (2003), Mars Exploration Rover Project Software Interface Specification (SIS) Camera Experiment Data

Record (EDR) and Reduced Data Record (RDR) Operations Data Products, *JPL Doc. D-22846*, Jet Propul. Lab., Pasadena, Calif.

Alexander, D., P. Zamani, R. Deen, P. Andres, and H. Mortensen (2005), Automated generation of image products for Mars Exploration Rover mission tactical operations, paper presented at 2005 International Conference on Systems, Man, and Cybernetics, Inst. of Electr. and Electron. Eng., Waikoloa, Hawaii.

Bell, J. F., III, et al. (2003), Mars Exploration Rover Athena Panoramic Camera (Pancam) investigation, *J. Geophys. Res.*, *108*(E12), 8063, doi:10.1029/2003JE002070.

Deen, R. G. (2003a), Cost savings through Multimission code reuse for Mars image products, paper presented at 5th International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations, Deep Space Commun. and Navig. Syst. Cent. of Excellence (DASCANSO), Jet Propul. Lab., Pasadena, Calif.

Deen, R. G. (2003b), Issues with linearization, *MER Docushare ID 75670*, Jet Propul. Lab., Pasadena, Calif.

Deen, R. G., and J. J. Lorre (2005), Seeing in three dimensions: Correlation and triangulation of Mars Exploration Rover imagery, paper presented at 2005 International Conference on Systems, Man, and Cybernetics, Inst. of Electr. and Electron. Eng., Waikoloa, Hawaii.

Gennery, D. B. (2001), Least-squares camera calibration including lens distortion and automatic editing of calibration points, *Calibration and Orientation of Cameras in Computer Vision*, edited by A. Grun and T. Huang, chap. 5, pp. 123–136, Springer, New York.

Gennery, D. B. (2002), Generalized camera calibration including fish-eye lenses, *JPL Doc. Clearance 03-0869*, Jet Propul. Lab., Pasadena, Calif.

Goldberg, S. B., M. W. Maimone, and L. Matthies (2002), Stereo vision and rover navigation software for planetary exploration, paper presented at 2002 IEEE Aerospace Conference, Inst. of Electr. and Electron. Eng., Big Sky, Mont.

Herkenhoff, K., et al. (2006), Overview of the Microscopic Investigation during Spirit's first 450 sols in Gusev crater, *J. Geophys. Res.*, doi:10.1029/2005JE002574, in press.

Huang, T. (2003), Component architecture: The software architecture for mission requirements, paper presented at 5th International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations, Pasadena, Calif.

Johnson, J. R., et al. (2006), Spectrophotometric properties of materials observed by Pancam on the Mars Exploration Rovers: 1. Spirit, *J. Geophys. Res.*, doi:10.1029/2005JE002494, in press.

Kochan, S. G., and P. H. Wood (1998), *Unix Shell Programming*, revised edition, Hayden, Carmel, Indiana.

LaVoie, S. K., et al. (1999), Processing and analysis of Mars Pathfinder science data at the Jet Propulsion Laboratory's Science Data Processing Systems Section, *J. Geophys. Res.*, *104*(E4), 8831–8852.

Leger, C., R. G. Deen, and R. G. Bonitz (2005), Remote Image Analysis for Mars Exploration Rover Mobility and Manipulation Operations, paper presented at 2005 International Conference on Systems, Man, and Cybernetics, Inst. of Electr. and Electron. Eng., Waikoloa, Hawaii.

Li, R., K. Di, L. H. Matthies, R. E. Arvidson, W. M. Folkner, and B. A. Archinal (2004), Rover localization and landing-site mapping technology for the 2003 Mars Exploration Rover mission, *Photogramm. Eng. Remote Sens.*, *70*(1), 77–90.

Maki, J. N., et al. (2003), Mars Exploration Rover Engineering Cameras, *J. Geophys. Res.*, *108*(E12), 8071, doi:10.1029/2003JE002077.

Maxwell, S., B. Cooper, F. Hartman, J. Wright, J. Yen, and C. Leger (2005), The best of both worlds: Integrating textual and visual command interfaces for Mars rover operations, paper presented at 2005 International Conference on Systems, Man, and Cybernetics, Inst. of Electr. and Electron. Eng., Waikoloa, Hawaii.

Neuman, B. C., and T. Ts'o (1994), Kerberos: An authentication service for computer networks, *IEEE Commun. Mag.*, *32*(9), 33–38.

Norris, J. S., M. W. Powell, M. A. Vona, P. G. Backes, and J. V. Wick (2005), Mars Exploration Rover operations with the Science Activity Planner, paper presented at 2005 IEEE Conference on Robotics and Automation, Inst. of Electr. and Electron. Eng., Barcelona, Spain.

Olson, C. F., L. H. Matthies, M. Schoppers, and M. W. Maimone (2001), Stereo Ego-motion improvements for robust rover navigation, paper presented at 2001 International Conference on Robotics and Automation, IEEE Robotics and Autom. Soc., Seoul, Korea.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling (1988), *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge Univ. Press, New York.

Soderblom, J. M., J. F. Bell, R. E. Arvidson, J. R. Johnson, M. J. Johnson, and F. P. Seelos (2004), Mars Exploration Rover Pancam photometric data QUBs: Definition and example uses, *Eos Trans. AGU*, *85*(47), Fall Meet. Suppl., Abstract P12A-0198.

Wright, J., A. Trebi-Ollennu, F. Hartman, B. Cooper, S. Maxwell, J. Yen, and J. Morrison (2005), Terrain modelling for in-situ activity planning and rehearsal for the Mars Exploration Rovers, paper presented at 2005 International Conference on Systems, Man, and Cybernetics, Inst. of Electr. and Electron. Eng., Waikoloa, Hawaii.

Yakimovsky, Y., and R. Cunningham (1978), A system for extracting three-dimensional measurements from a stereo pair of TV cameras, *Comput. Graphics Image Process.*, *7*, 195–210.

D. A. Alexander, P. M. Andres, M. K. Cayanan, A. C. Chen, R. G. Deen, J. R. Hall, V. S. Klochko, H. B. Mortensen, O. Pariser, C. L. Stanley, C. K. Thompson, G. M. Yagi, and P. Zamani, Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109, USA. (doug.alexander@jpl.nasa.gov)